ReportBuilderPro
Interim Report

TU857
BSc in Computer Science (Infrastructure)

Student Name: Derry Mahon
Student Number: C22445282

Supervisor: Deirdre Lawless

School of Computer Science
Technological University, Dublin

Date: 20/10/2025

Abstract

ReportBuilderPro will be an integrated digital reporting system developed specifically for the AEC Sector. It will consist of both a web and mobile application, designed to streamline progress reporting, health and safety documentation and risk identification through real time data capture and AI enhanced analysis. The system will address key challenges in current workflows, such as manual data entry, inconsistent reporting formats, and poor connectivity on remote sites.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

_____

Derry Mahon

Date 22/11/2025

Acknowledgements

I want to thank Deirdre Lawless, my project supervisor, for all the expert advice and valuable insights she gave me during this project. She has been extremely supportive in all aspects of the project for which I am extremely grateful. I also want to thank all the MGM Partnership employees who contributed. Their feedback was crucial for the system's development and evaluation. I really appreciate all the help these people provided me on the project.

Contents

1. Introduction

1.1 Project Background

The Architecture, Engineering, and Construction (AEC) sector in Ireland is rapidly changing due to innovations designed to enhance efficiency throughout the project lifecycle. The Irish Governments *Build Digital Project Initiative 2030* captures this stating *"We believe that through the adoption of digital practices, the Irish construction and built environment sectors will see a transformation in the way we work. Knowledge sharing, information management and transfer will become more streamlined"* [1]. Despite this many companies still rely on outdated methods for critical tasks such as project tracking and reporting. These outdated methods cause delays, mistakes, and missed chances for a more proactive and streamlined approach, which raises costs and lowers productivity.

I have experienced these issues in person while working for a Chartered Project Management and Quantity Surveying firm. Manual reporting methods often involve fragmented communication between the site and the office, particularly in remote or newly developed areas where connectivity is often poor. Deadlines are often pushed back due to a lack of standardised templates and manual compilation of images, notes, and observations. this is not only evident in my own experience but is consistently highlighted as a systemic issue across the AEC Sector [2]. The typical workflow for construction project includes numerous stages from data collection to visualisation of progress in reports (see Figure 1). These operational bottlenecks not only slow down project delivery, but they also raise costs due to the increased man hours required in report compilation.

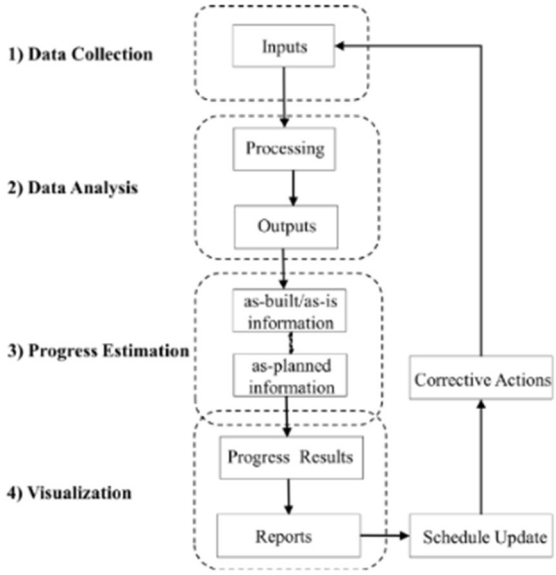


Figure 1 Overview of the current flow of Progress Reporting [3]

The integration of digital tools particularly AI and mobile technologies presents a significant opportunity to streamline these workflows.[4] Companies are looking into AI and other technologies more and more to see if they can automate and improve construction monitoring and reporting. Even though we can never get rid of the human element in an industry run by people, technology is always changing to make the industry more accurate and efficient. This is summarised well by Dan Hughes of Alpha Property Insight: *"Technology is really good at number crunching, and then the human will do what the human is good at—ethics, judgement, interpretation, human interaction."*[5]. His opinion confirms the idea that technology should drive construction reporting, but people should be in charge.

A recent report from the Society of Chartered Surveyors, the leading body of property, land, and construction professionals in Ireland titled *"Will AI help quantity surveying?"* [6]. This feature, written by Mary C. Flynn, Assistant Chief Quantity Surveyor at Dublin City Council, explores the use of AI tools and how they are reshaping the construction industry. This article also quotes Kathy Bloomgarden, CEO of Ruder Finn endorsing the idea of optimising workflows in the industry: "*We are in a moment of unprecedented human potential, and AI is the key to unlocking that potential*." [6]. These perspectives highlight the growing consensus that digital transformation is essential for future proofing the AEC sector.

In my current role in the construction sector, My responsibilities include leveraging technology to improve workflows and efficiency in a sector which lags behind others in technological adaption [7]. I focus on exploring and implementing digital solutions, particularly AI driven tools, to enhance reporting and project management processes. In addition to this technology focused work, I also assist with site visits giving me a practical insight into the challenges faced on site, such as fragmented communication and delays caused by manual reporting. This combination of technical knowledge and on-site experience inspired me to develop ReportBuilderPro an integrated web and mobile application that combines mobile accessibility, AI driven insights, and real-time data synchronisation. By addressing these challenges, the platform aims to deliver accurate, user-friendly reporting while remaining adaptable across diverse AEC contexts. Incorporating user feedback and iterative testing will ensure the solution is grounded in practical needs and capable of driving meaningful digital transformation in the industry. Despite the availability of digital tools, existing solutions often fail to integrate offline first reporting, AI driven analysis, and standardised templates within a single tool. This creates a gap for a unified, intelligent reporting application designed for real-world construction site constraints.

1.2 Project Description

ReportBuilderPro will be an integrated digital reporting system developed specifically for the construction industry. It will consist of both a web and mobile application, designed to streamline progress reporting, health and safety documentation and risk identification through real time data capture and AI enhanced analysis. The system will address key challenges in current workflows, such as manual data entry, inconsistent reporting formats, and poor connectivity on remote sites.

The Web Application aspect will serve as the central control hub for report template creation, refinement, project oversight and client deliverables. It will feature a dashboard which will display data taken from our report scans that allows users to monitor project risks, timelines and possible shortages of materials. Reports which are submitted from the field will be refined and can be ran through our AI scanner which will be able to classify and highlight potential issues in a visual dashboard, providing insights and supporting proactive decision making before generating a professional PDF output for client delivery.

The Mobile Application will be tailored for on-site usage by project managers, quantity surveyor or other expert personnel. It will support offline-first data capture, enabling users to document progress, upload images, dictate notes and complete safety checklists even in areas with limited connectivity. Once online, the app will sync with the backend to update the report for further completion on the web application.

One of the most important features in ReportBuilderPro will be its integration of AI technologies to enhance reporting intelligence. The system will use Natural Language Processing (NLP) to automatically scan report text for risks, delays, and material shortages.

Computer vision models will assess uploaded images to detect safety hazards and progress indicators. These AI tools will flag issues, identify patterns across projects, and provide smart recommendations to users, such as highlighting missing information or suggesting corrective actions. The AI models will improve over time through continuous learning, ensuring that insights become more accurate and context-aware with each use. The system will be driven by technology but led by people. This will help professionals make decisions more quickly and with more information, while still allowing them to control how they interpret and judge things.

1.3 Project Aims and Objectives

The goal of the ReportBuilderPro is to create an integrated digital reporting system consisting of a web app and a mobile app that makes it easier, more accurate, and more accessible to report on progress, health and safety, material shortages and risks in the construction industry. The system will use AI-driven analysis and offline-first mobile data capture to make reporting workflows easier and help people make decisions based on data in a timely manner.

The scope of this project will focus on reporting workflows within construction site operations, specifically progress tracking, health and safety documentation, and material shortage identification. It will not extend to full project management or financial planning modules.

Project Objectives

Conduct a comprehensive review of existing technologies to identify current capabilities, limitations and opportunities for innovation particularly in the AI integration and offline functionality.

Conduct Expert Elicitation through questionnaire as well as direct observation of real-world workflows to ensure the system addresses practical, on site and in office needs

Appropriate System Architecture that separates the web application, mobile application, backend services and AI component, ensuring scalability maintainability and ease of integration

Develop a web application that enables users to create and refine customisable report templates, visualise flagged risks, delays and material shortages via dashboard

Develop a mobile application that allows onsite users to access and complete report templates, capture images dictate notes and operate offline with automatic synchronisation upon reconnection

Implement a robust backend using appropriate technologies to manage data storage user authentication and file handling efficiently and securely

Integrate a Natural Language Processing component to analyse report text, classify content into relevant categories such as delays, risks, and material shortages, and flag potential issues within the system's dashboard adopting the technologies most appropriate for the project.

Evaluate the system through functional, usability, and performance testing using real user feedback to assess accuracy, ease of use, offline performance, and the quality of generated

reports. Additionally, evaluate the usefulness of the system using feedback from at least three domain experts to ensure practical relevance and industry alignment.

1.4 Project Scope

ReportBuilderPro focuses on the design and development of a digital reporting system for the construction industry, comprising of a web application and a mobile application. The scope includes core functionalities such as template creation, offline first data capture, AI driven report analysis and PDF generation. The system is intended for use by Quantity surveyors and Project managers and other site personnel to streamline progress reports, health and safety documents and risk identification. Key constraints include internet connectivity on construction sites, varying levels of digital literacy among users, and the need for AI outputs to serve as advisory rather than authoritative recommendations.

What is in the Project Scope?

- Development of a web application for report template creation, dashboard monitoring, and PDF exporting.
- Development of a mobile application for on-site data entry, including image uploads and voice notes.
- Implementation of offline-first functionality for mobile users.
- Integration of a Natural Language Processing (NLP) engine to analyse report text and flag risks, delays, and material shortages.
- Setup of a backend infrastructure for data storage, user authentication, and file handling.
- Functional, usability, and performance testing using real user feedback from a construction company environment.
- Delivery of system documentation, architecture diagrams, and a final evaluation report.

What is out of Project Scope

- Full-scale deployment across multiple companies or commercial environments.
- Integration with third-party enterprise systems (e.g., ERP, BIM platforms).
- Real-time collaboration features (e.g., multi-user editing or live chat).
- Support for regulatory compliance across different countries or regions.
- Mobile app publication to public app stores (e.g., Google Play, Apple App Store).
- Long-term maintenance, updates, or scalability planning beyond the prototype phase.

1.5 Thesis Roadmap

1. **Introduction**: Discusses the projects background, motivation, aims, and objectives, and outlines the scope of the work, providing a roadmap for the thesis.

2. **Literature Review**: This chapter explores existing solutions and research relevant to construction reporting tools and similar applications. It will also it will examine technologies, methodologies for the technical aspects of this project along with their strengths and limitations.

3. **System Analysis**: Analyses system requirements and architectural decisions, explaining the chosen software methodology and how components interact to meet user needs.

4. **System Design**: Provides an in-depth look at the architectural and design choices,

explaining the selected software methodology, system components, and how each part of the application will interact to deliver a cohesive user experience.

5. **Testing and Evaluation**: Details the comprehensive testing strategy, including unit, integration, and user testing, and describes the methods used to evaluate the project's performance, usability, and reliability.

6. **Prototype Development**: Documents the development of the initial prototype and the current state of the project, this chapter will also include code snippets and screenshot of the current working version.

7. **Issues and Future Work**: Reviews the project's current limitations, potential risks, and outlines areas for future development that need to be completed for achieve the end goal of this project.

2. Literature Review

2.1 Introduction

This section reviews existing literature, technologies, and systems relevant to the development of a digital reporting solution for the construction industry based upon my research conducted in Section 1. It explores current software tools used in the field, evaluates the technologies selected for implementation, and examines domain-specific research on construction reporting and AI integration. It also considers similar final-year projects to position this work within the academic context and identify areas for improvement.

The initial concept originated from my personal experience in the construction industry; however, it became evident that the challenges I identified, including inefficiencies, disjointed workflows, and a scarcity of user-friendly digital tools, were widespread. These issues are repeated across the industry and are well documented in EY's *Detailed Description of Needs for the Irish Construction/Built Environment Sector* [8] under Project Ireland 2040. Ireland's long-term, overarching strategy for social, economic, and cultural development, combining the National Planning Framework (NPF) [9] and the National Development Plan (NDP)[10] to guide the country's future to 2040

The report shows the difference between companies that are open to new ideas and those that aren't. For instance, *"The rapid pace of technological change is already seeing some early adopters in the construction industry adopt new technological processes and more advanced construction systems to deliver value to their clients."* [8]. It also notes *"40% of firms in the industry are not using any form of automated technologies within their current projects."* [8]. This gap makes it even more important to have a user-friendly system that makes it easier for small and medium-sized businesses to get started, encourages digital adoption, and lets people work together across the supply chain.

2.2 Alternative Existing Solutions

As part of my research, I looked into the main technologies that are currently driving the Irish construction industry as part of my research. These tools are actively employed by some of the country's leading firms to deliver multi-million-euro developments and play a vital role in supporting every phase of the project lifecycle.

Tool 1: Procore

Procore is a full-featured construction management platform that lets you document projects, set budgets, make schedules, and work together. It is one of the most popular tools in the AEC industry.

URL: https://www.procore.com

- *Strengths*: Procore offers an integrated suite of project management tools, strong mobile support and seamless cloud accessibility, enabling efficient coordination across multiple stakeholders.
- *Weaknesses*: However, the platform is relatively expensive has limited offline capabilities and presents a complex user interface that may not be well-suited to smaller teams or firms with limited technical experience.
- *Relevance*: ReportBuilderPro will make it easier for smaller teams to access reports and make them easier to read.

Tool 2: SiteCam

A mobile-first site documentation tool that lets you take pictures and write notes to keep track of your progress.
URL: https://sitecam.io/

- *Strengths*: SiteCam excels in image-based reporting and offers a user-friendly mobile interface, making it ideal for quick on-site documentation.
- *Weaknesses*: The platform lacks advanced template customization and does not support AI-driven analysis, which limits its scalability for more complex reporting needs.
- *Relevance*: ReportBuilderPro will greatly improve SiteCam's capabilities by adding AI-powered risk detection and smart reporting workflows. Its flexible templates and structured data capture make it perfect for teams that need more than just visual documentation, especially when safety and compliance are very important.

Tool 3: PowerProject

Power Project is used for scheduling, managing of resources, and track the progress of construction projects widely used for planning purposes.
URL: https://www.elecosoft.com/products/powerproject

- *Strengths:* Offers advanced Gantt chart scheduling, robust resource management, and BIM integration. It is well-established in the industry for planning and tracking project timelines.
- *Weaknesses:* Not designed for health and safety reporting or on-site data capture. Lacks mobile-first functionality and does not support AI driven analysis or automated reporting workflows.
- *Relevance*: PowerProject is great at planning and scheduling, but it doesn't have the tools for field-level reporting and safety oversight. ReportBuilderPro will fill this gap by providing real-time, AI-enhanced reporting, the ability to work offline, and smart risk flagging that sends actionable insights straight from the site to decision makers.

Tool 4: Fieldwire

Fieldwire is a site management solution developed for the field, enabling you to share the right information with your teams and coordinate on-site in real time.
URL: https://www.fieldwire.com

- *Strengths*: Offers robust task tracking and drawing management features, enabling efficient communication and collaboration among field teams.
- *Weaknesses*: Lacks dedicated functionality for health and safety reporting, automated reporting workflows, and customizable templates, which can limit its effectiveness for compliance-driven projects.
- *Relevance:* ReportBuilderPro will fill this gap by providing structured, AI-powered reporting and safety compliance tools that work automatically. Its smart templates and real-time insights make it easy for small teams to write high-quality reports, which improves both accuracy and decision-making on the job site.

Comparative analysis of my System vs Industry Systems

Feature	ReportBuilderPro	Procore	SiteCam	Powerproject	Fieldwire
Offline-first data capture	Yes	No	Yes	No	Yes
AI issue finder	Yes	No	No	No	No
PDF report generation	Yes	Yes	Yes	No	Yes
Template customization	Yes	Yes	No	No	Yes
Real-time dashboard	Yes	Yes	No	Yes	Yes

2.3 Technologies Researched

This section outlines the technologies selected for the development of my reporting system. I chose each part based on how well it would work for quick prototyping, how well it would work for scaling, and how well it would work with the system's functional needs, especially offline first mobile reporting and AI-enhanced analysis.

2.3.1 Frontend Technologies

"In the dynamic realm of web development, the front-end serves as the gateway to user interaction, making the selection of appropriate technologies a pivotal decision for developers and organizations." [11]

To support the goals of Report Builder Pro, the frontend must deliver a responsive, modular interface for building and managing reports, with smooth user interactions and real-time updates. These are the front-end technologies I researched.

React.js: *(Selected) URL https://react.dev/*

A widely adopted JavaScript library for building dynamic user interfaces. Reacts component-based architecture supports modular design and efficient rendering, making it ideal for building the dashboard and report template builder in the web application.

Vue.js:

A progressive JavaScript framework known for its simplicity and ease of integration. Vue is lightweight and used mainly in smaller projects

Angular:

A full-featured frontend framework developed by Google, offering built-in routing, state management, and testing tools.

I will use React.js for the front end of Report Builder Pro because its component-based architecture makes it easy to make dynamic, reusable UI elements that are perfect for making an interactive report template builder and dashboard. React was chosen because it has good documentation, active communities, and can help with quick development in the short time frame of a dissertation project.

2.3.2 Backend Technologies

The backend must support secure user authentication, scalable processing of diverse data types including text, images, and structured inputs and efficient communication with the NLP engine. As technology continues to advance, backend innovation will be pivotal in supporting

sustainable, secure, and intelligent digital infrastructure.[12] Particularly for ReportBuilderPro the importance of reliability and security is of the upmost importance.

Node.js: *(Selected) URL https://nodejs.org/en*

A lightweight and scalable runtime environment for building server-side applications. It will be used to manage user authentication, report submission, and communication between the frontend and backend.

FastAPI: *(Selected) URL https://fastapi.tiangolo.co*

A modern Python framework for building APIs, chosen specifically for integrating the NLP engine. FastAPI supports asynchronous operations and is optimized for performance.

Django:

A high-level Python web framework with built-in admin tools and ORM support. It's well-suited for traditional web applications.

Flask:

A lightweight Python micro-framework used for building simple web applications and APIs.

For the backend of Report Builder Pro, I will use Node.js to handle user authentication and manage communication between the frontend and backend efficiently, while FastAPI is selected for its high performance and compatibility with Python-based NLP tools, making it well suited for integrating the natural language processing engine. These technologies were chosen for their strong documentation, active communities, and ability to support rapid development within the limited timeframe of a dissertation project.

2.3.3 Database

ReportBuilderPro will require a database solution that can handle a wide variety of data types, including structured user information, unstructured report content, and AI generated analysis results. Flexibility in schema design and scalability are key, given the diverse nature of the data being processed during report generation.

MongoDB: *(Selected) URL https://www.mongodb.com/*

A NoSQL document-oriented database that supports flexible schema design. It is well-suited for storing varied report formats, user data, and AIanalysis results.

PostgreSQL:

A powerful relational database known for its reliability and support for complex queries

Firebase:

A cloud-hosted NoSQL database optimized for mobile apps with real-time syncing.

MongoDB will be the main database for ReportBuilderPro because its document-oriented structure makes it easy to store different types of reports and NLP outputs. Its ease of integration with Node.js and support for rapid development made it the most suitable choice for ReportBuilderPro.[13]

2.3.4 AI and NLP Tools

To enable intelligent analysis of construction reports, Report Builder Pro will incorporate natural language processing (NLP) and machine learning tools that will extract insights from unstructured text. These tools will be used to identify risks, delays, and material shortages based on patterns in user-generated content.

spaCy: *(Selected) URL https://spacy.io/*

A robust Python library for natural language processing, used for text preprocessing such as tokenization, entity recognition, and sentence parsing.

Scikit-learn: *(Selected) URL https://scikit-learn.org/stable/*

A machine learning library used to train and evaluate classification models that flag risks, delays, and material shortages in report text.

NLTK:

A classic Python library for natural language processing, widely used in academic settings.

PyTorch:

Deep learning frameworks used for building complex AI models.

spaCy will be selected for its efficient text preprocessing capabilities, including tokenization and entity recognition, while Scikit-learn will be used to train lightweight classification models that flag key issues in report text. These libraries will support rapid experimentation and integration with the Python-based backend.

2.3.4 PDF Generation

To produce professional, client-facing reports, Report Builder Pro will require a solution for converting structured HTML templates and AI enhanced content into downloadable PDF documents.

pdfkit: *(Selected) URL https://pdfkit.org/*

A Python library used to convert HTML content into PDF format. It will be used to generate professional client-facing reports from the refined templates and AIanalysis results.

pdfkit will be used to handle this conversion, offering a simple and effective way to generate consistent, well-formatted PDFs directly from the backend.

2.3.5 Cloud Hosting Services

Cloud services will be used to host the application and store user-generated content, including images and reports. While long-term scalability is not a requirement, reliable deployment and secure storage will be essential during development and testing.

AWS (Amazon Web Services): *(Selected) URL* https://aws.amazon.com/

Used for secure file storage, particularly for images and report documents uploaded via the mobile app.

Azure: *(Selected) URL https://azure.microsoft.com/en-us/*

Selected for hosting the web application and backend services due to its integration with development tools and scalability options.

Azure will be chosen to host the web application and backend services because it works well with development tools and is easy to set up. AWS will be used to store uploaded files, especially images and PDFs, because it has strong and safe storage options.

2.4 Other Research

2.4.1 Expert Elicitation

Conducting an elicitation was a key part in finalising my system requirements. It allowed me to gather informed insights from both field experts and real users ensuring the system was grounded in the practical needs of the user. One particularly useful framework to this was *A*

practical guide to structured expert elicitation using the IDEA Protocol by Victoria Hemming et al[14].

The IDEA Protocol (Investigate–Discuss–Estimate–Aggregate) widely used for its ability to reduce bias and improve the reliability of expert input.

Before applying the protocol, I conducted essential pre-elicitation steps, which included:

- Compiling relevant background information to inform participants
- Identifying and contacting suitable experts across construction and technology domains
- Briefing participants on the elicitation process, its purpose, and how their input would shape the system design.

These steps ensured that the elicitation was well-informed and that participants were prepared to contribute meaningfully. The structured nature of the IDEA Protocol helped me refine system features such as reporting workflows, risk detection mechanisms, and template flexibility, aligning them with real-world expectations and industry standards. As stated by Hemming in the piece, *"The protocol is designed to reduce overconfidence, encourage diversity of opinion, and improve the accuracy of group judgements."* [14]. Highlighting the benefit to using IDEA over informal consultation to ensuring my finalised system requirements met user needs.

Overview of the IDEA Protocol [14]

The IDEA protocol stands for:

Investigate: All experts individually answer questions and provide reasoning on their judgement.

Discuss: Experts share reasoning and insights to understand differing views.

Estimate: Experts revise their judgments based on the discussion.

Aggregate (Post-elicitation): Final judgments are combined to form a collective view.

This protocol is designed to reduce bias, improve the quality of expert input, and ensure transparency in decision-making. It is particularly useful when designing systems that rely on domain-specific knowledge, like ReportBuilderPro does.

How I applied IDEA Protocol

Investigate

I began by identifying key stakeholders in my userbase such as Project Managers and Quantity Surveyors. Each expert was asked to complete a questionnaire individually, focusing on their current reporting practises, pain points, and desired features.

Discuss

After I collected the initial responses, I followed up with informal interviews and emails to further clarify the data I had collected. This stage helped me gain deeper insights to the workflows challenges and prioritise features

Estimate

Based on the discussions, I asked the experts to revisit initial feedback and refine their input. This iterative feedback helped me shape a more accurate and prioritised set of requirements

Aggregate

I complied my results into a consolidated requirements list, identifying common themes and critical needs. These aggregated insights directly informed me of the system specifications and design decisions

How it strengthened my Project?

- The IDEA Protocol added rigor to my requirements gathering process.
- It ensured the system was grounded in the real experts needs and not based off assumption
- It provided me with a defensible methodology for stakeholder's engagement which strengthened the credibility of this project

2.4.2 Feature Driven Development (FDD)

To guide the development of my system, I adopted a Feature-Driven Development approach. FDD is a client centric, iterative approach that focuses on designing and building software around clearly defined user valued features. This approach aligns well with my project goals.

'Feature Driven Development (FDD) and Agile Modeling'. An article by Scott W. Ambler an international keynotes speaker [15]. Outlines the methodology as built on five key activities:

1. Develop an overall model.
2. Build a feature list.
3. Plan by feature.
4. Design by Feature.
5. Build by feature.

By implementing this structure, it ensures the system is built in both systematic and responsive to the user's needs.

Overview of FDD Principles[15]

- **Feature-centric** Each feature represents a small, client-recognizable piece of functionality
- **Iterative and incremental Features** are developed in short cycles, allowing for continuous refinement.
- **Domain-driven** The system is modelled around real-world entities and processes, such as reports, users, and safety flags.
- **Team-oriented** Roles such as Feature Owner and Class Owner help ensure accountability and collaboration.

As stated in the reference from Ambler, *"FDD blends several industry-recognized best practices into a cohesive whole. These practices are driven from a client-valued functionality (feature) perspective."* [15]

Applying FDD to my Project

Develop an overall model to implement Feature-Driven Development (FDD) for ReportBuilderPro, I began by mapping the system's core entities users, reports, templates, risk flags, and workflows. This initial model was informed by expert elicitation results, gathered using the IDEA Protocol outlined in Section 2.41.
Build a Feature List drawing on the elicitation outcomes, I compiled a comprehensive list of desired features to be incorporated into the final system model.

Plan by Feature the features were prioritised based on user feedback and project feasibility. These were then organised into logical modules to ensure a structured development approach.

Design by Feature each feature was designed with a strong emphasis on user experience—focusing on simplicity, accessibility, and alignment with real-world workflows.

Build by Feature the system architecture and design decisions were structured to support modular, feature-based development.

2.5 Existing Final Year Projects

As part of this research, I accessed the university library to review previous projects, focusing on work that demonstrated advanced mobile and web development, innovative applications of machine learning, progress tracking or reporting systems and projects that incorporated complex or multi database architecture.

*Note all past projects were accessed from the TU Dublin Library Catalogue [16]

2.5.1 Project One

Title: Automatic License Plate Detection using Image Processing

Student: Ayan Abedin

Description (brief): The project aims to develop a robust and efficient system for real-time Automatic License Plate Recognition (ALPR) using OpenCV, Python programming, and advanced computer vision techniques. The primary objective is to create a comprehensive solution that automates the identification and processing of license plates within parking management systems. The system uses a computer/Raspberry Pi and a camera for image capture, employing OpenCV and Python for computer vision algorithms. It follows stages of image capture, quality enhancement, identifying license plate areas, character segmentation, and recognition, prioritizing real-time processing on the Raspberry Pi.

What is complex in this project:

The creation of a real time image recognition system which I could implement for risk delays and material shortage based off trends and my report and images.

What are the key aspects and complexity in this project?

What's described in the complexity is a key strength and selling point of this project however this is already a widely used and accessible technology already in use by An Garda Siochana for example.

2.5.2 Project Two

Title: Anseo [17]

Student: Johnathan Hew

Description (brief):

The goal of this project is to create an efficient, user-friendly solution that not only addresses the limitations of the pen and paper method but also offers additional features, such as reporting and data visualization. These added functionalities enable educators to better understand attendance trends and make informed decisions to improve student engagement and performance. To achieve this, the Anseo application will be developed using PostgreSQL, Express, React and Node using Feature Driven Development (FDD), an agile framework that focuses on delivering features incrementally and adapting to feedback throughout the development process. By implementing Anseo, institutions can streamline the attendance management process and leverage data-driven insights to enhance the overall educational experience.

What is complex in this project:

- Data Visualisation and Reporting
- Real Time Data Handling
- Scalability and Performance
- Feature Driven Design

What technical architecture was used:

- Postgres SQL Relational database for storing attendance records and other user data
- Express.js backend framework for API requests and server-side logic
- React Frontend Library for dynamic user responsive interface
- Node.js Runtime environment for executing JavaScript on server

What are the key strengths and weaknesses associated with this project?

User centric design, Data Drive insights and Modern tech stacks are clear strengths in this project and will be implemented in mine too.Some weaknesses may be the complexity in Visualisation which could misalign the priorities of the features if not careful

2.5.3 Project Third

Title: Deep [18]

Student: Bryan McCarthy

Description (brief):

The “Deep” project is a web application designed to enhance productivity and time management by minimizing distractions and promoting focused work sessions. It targets university students and incorporates tools like task management, calendar scheduling, Pomodoro timers, and data analytics to help users track and improve their study habits. The application is built using a modern tech stack including TypeScript, React.js, Vite, SASS, Go, Gin, GORM, and PostgreSQL. It follows Agile development with a Scrum framework to ensure iterative progress and adaptability to feedback.

What is complex in this project?

- Real-Time Task Tracking and Estimation: Estimating time remaining based on difficulty, progress, and time spent.
- Data Visualization: Dashboard analytics showing time allocation and milestones.

- User Experience Design: Minimalist, distraction-free interface with responsive design.
- Agile Development with Scrum: Managing iterative development and backlog prioritization.
- Session Management and Secure Routing: Using Gorilla sessions and Gin framework for secure backend operations.

What technical architecture was used:

- Frontend: React.js and Vite a responsive and component-based UI.
- Backend: Go with Gin framework for routing and GORM for ORM.
- Database: PostgreSQL for storing user data and task records.
- Session Management: Gorilla/sessions for secure cookie-based sessions.
- Local Storage: Used for storing less critical data like timer settings.

What are the key strengths and weaknesses associated with this project:

Strengths:

- User-Centric Design: Clean, minimalist interface focused on reducing distractions.
- Comprehensive Productivity Tools: Task management, calendar, Pomodoro timer, and dashboard analytics.
- Modern Tech Stack: Efficient and scalable technologies with strong community support.
- Agile Methodology: Scrum framework allows for iterative development and responsiveness to feedback.
- Strong Conceptual Foundation: Inspired by Cal Newport's "Deep Work" and GTD methodology, aligning features with cognitive science.

Weaknesses:

- Complexity in Estimating Time Remaining: The algorithm for predicting task completion time may need refinement.
- Potential Overhead in Tech Stack: Using Go and Gin may be overkill for simpler applications unless scalability is a priority.
- FDD Not Used: Unlike Anseo, this project uses Scrum, which may be more flexible, but less feature focused.

2.6 Conclusions

Based on my research ReportBuilderPro should be designed as scalable, intelligent reporting solution for construction and project environments, built on a carefully selected tech stack that will balance performance, usability and extensibility. Using spaCy for preprocessing and Scikit-learn for model development ensures robust natural language processing. Fast API acts as the API layer that wraps the NLP to the rest of the system serving as the bridge between frontend and backend while PostgreSQL offers a structured, relational database environment with strong data integrity and advanced querying capabilities—especially well-suited for analytics and reporting workflows. React powers the dashboard for dynamic, user-friendly interaction with the data. Hosted on Azure, the platform benefits from enterprise-grade

reliability and scalability. Development follows Feature Driven Development (FDD), allowing incremental delivery of high-value features. Requirements were elicited through analysis of existing reporting workflows, user interviews, and benchmarking against tools like SiteCam, Procore, Fieldwire and ProjectPlanner ensuring the solution is grounded in real-world needs and optimized for construction reporting challenges.

3. System Analysis

3.1 System Overview

ReportBuilderPro is an web and mobile application designed to streamline construction site reporting. It enables site managers and engineers to quickly create, manage, and export structured reports using natural language input. With a user-friendly dashboard, real-time data visualization, and automated PDF generation, the system enhances productivity and ensures consistent, high-quality documentation.

3.2 Stakeholder and User Analysis

3.2.1 Stakeholder Identification

Stakeholder identification is critical to the functional success of software development projects [19] to ensure the systems design reflects the real-world needs and expectations. This is particularly relevant for ReportBuilderPro, given its user-centric design approach, which prioritises the requirements and workflows of construction professionals to deliver a practical and intuitive solution.

During those initial stages I identified the following stakeholder groups

Primary Stakeholders

- Project Managers
- Quantity Surveyors
- Company Directors
- IT Admin

Secondary Stakeholders

- Clients

To ensure my system met the needs of real-world experts I carried out expert elicitation using the IDEA Protocol [14] as outlined in Section 2.4.1 of this report.

In addition to a formal elicitation, I also drew on

- Firsthand experience with current workflows.
- Competitor analysis to benchmark features and identify gaps

3.2.2 Stakeholder Needs

Each stakeholder group has distinct priorities and expectations. Based on my expert elicitation and follow up conversations the main needs of stakeholders were identified as follows

- **Project Managers** Require real-time progress reporting, risk identification, and streamlined health and safety documentation.
- **Quantity Surveyors** Need accurate material tracking and cost-related reporting integrated into site workflows.
- **Company Directors** Seek high-level summaries and compliance assurance for decision-making.
- **IT Administrators** Focus on system security, data integrity, and ease of deployment.

3.2.3 Workflow Pain Points

Current reporting workflows in construction have problems that make it harder to be efficient and follow the rules. This section talks about these problems, which are the main objectives for the design of ReportBuilderPro.

- Manual compilation of images, notes, and observations leading to delays and miscommunication.
- Lack of standardised templates across projects, increasing risk of non-compliance.
- Intermittent connectivity on construction sites, making real-time reporting difficult.
- Limited integration of AI-driven insights in existing tools, resulting in missed opportunities for initiative-taking risk management.

3.3 Requirements Gathering and Elicitation

This section outlines the functional and non-functional requirements for the proposed site reporting system. The analysis is grounded in best practices from software engineering and user-centred design and considers emerging technologies such as natural language processing (NLP) to support intelligent reporting workflows in construction management.

3.3.1 Methods Used

The requirements gathering process for ReportBuilderPro was designed to be rigorous and stakeholder driven. To achieve this, I adopted the IDEA Protocol [14], a structured elicitation method that reduces bias and improves the accuracy of expert input.

Overview of the IDEA Protocol [14]

- **Investigate** All experts individually answer questions and provide reasoning on their judgment.
- **Discuss** Experts share reasoning and insights to understand differing views.
- **Estimate** Experts revise their judgments based on the discussion.
- **Aggregate** Final judgments are combined to form a collective view.

This protocol is particularly useful for systems like ReportBuilderPro that rely on domain specific knowledge, as it ensures transparency and diversity of opinion.

How I Applied the IDEA Protocol

- **Investigate** I began by identifying key stakeholders such as Project Managers and Quantity Surveyors. Each expert completed an individual questionnaire focusing on current reporting practices, pain points, and desired features.
- **Discuss** After collecting initial responses, I conducted informal interviews and email follow ups to clarify and deepen insights into workflow challenges.
- **Estimate** Based on discussions, experts revisited their initial feedback and refined their input, helping shape a more accurate and prioritised set of requirements.
- **Aggregate** I compiled results into a consolidated requirements list, identifying common themes and critical needs. These aggregated insights directly informed system specifications and design decisions.

How It Strengthened My Project

- **Added rigor** The IDEA Protocol provided a structured, defensible methodology for requirements gathering.
- **Grounded in real needs** Ensured the system was based on expert input rather than assumptions.
- **Improved credibility** Demonstrated a transparent and systematic approach to stakeholder engagement.

- **Clarity of Features** The data shows a clear trend as to where the workflow can be improved and provided insight on what ReportBuilderPro's functionality should be.

3.3.2 Key Findings

The elicitation process highlighted several critical requirements:

- **Standardised templates** to reduce inconsistencies and compliance risks.
- **Offline first functionality** Essential for sites with intermittent connectivity.
- **AI driven analysis** to classify delays, risks, and material shortages automatically.
- **User friendly interface** Designed for varying levels of digital literacy among site personnel.
- **Secure data handling** to protect sensitive project information during sync and storage.

3.4 Functional and Non-Functional Requirements

3.4.1 Functional Requirements

Structured Report Generation via Web Interface

- The system shall enable users to create site reports using a structured, simple interface.
- Justification: Structured input ensures consistency and facilitates data validation,

Natural Language Processing (NLP) for Report Analysis

- The system shall incorporate NLP techniques to analyse free text entries.
- Justification: Reduces cognitive load on users and enhances report readability; aligns with current trends in intelligent document processing.

PDF Export Capability

- Users shall be able to export reports in PDF format for offline access, sharing, and archival.
- Justification: PDF is a widely accepted format in professional and regulatory contexts.

Interactive Dashboard for Report Management

- The system shall provide a dashboard for viewing, filtering, and analysing reports.
- Justification: Enhances situational awareness and supports decision-making through visual analytics.

User Authentication and Role Based Access Control

- The system shall implement secure login and access control mechanisms based on user roles.
- Justification: Ensures data confidentiality and integrity; aligns with principles of least privilege.

3.4.2 Non-Functional Requirements

Performance

- NLP processing and dashboard interactions shall exhibit low latency

- Justification: High responsiveness is critical for user satisfaction and real-time decision-making.

Security

- Data shall be transmitted over HTTPS and stored in encrypted databases
- Justification: Protects sensitive information and ensures compliance with data protection regulations (e.g., GDPR).

Mobile Responsiveness

- The system shall be optimized for mobile devices to support on-site usage.
- Justification: Field operatives require access to reporting tools in dynamic, mobile environments.

3.4.3 Requirement Prioritisation (MoSCow)

The requirements matrix provides a clear link between functional requirements and user needs, helping prioritise development tasks. Each requirement was derived directly from the use case specifications, ensuring that the system design reflects real-world workflows and user goals.

Req ID	Name of Req	Description	Priority	User Contact
R!	Report Creation	Create and edit site reports	High	PM
R2	Image Tagging	AI-based photo content detection	Medium	PM
R3	Offline Sync	Save and sync data offline	High	PM
R4	PDF Export	Generate client-ready reports	High	QS and PM
R5	Dashboard View	Real-time project tracking	Medium	Admin
R6	NLP Risk Detection	Flag risks, delays, shortages	High	Admin

Figure 2 Requirements Specification Matrix

3.5 Use Case Analysis

Use cases are very important in software development because they give a structured way to get functional requirements from the user's point of view. They make sure that the design of the system is based on how things work in the real world, not just on technical features. *"The impact of use case diagrams on software development cannot be overstated. Their ability to simplify complex requirements, clarify system boundaries, and ensure user-centered design contributes directly to the success of software projects."*[20] . Use cases improve requirements elicitation, stakeholder communication, and system design clarity, making them essential to building user centred applications.

For ReportBuilderPro, use cases are particularly important because the system operates in a dynamic, high-pressure environment the construction site. By modelling tasks such as creating templates, documenting on site progress, submitting reports for AIanalysis, and reviewing insights, we ensure that the application supports actual user goals while remaining intuitive for users who may not be highly tech-literate. This method is in line with ISO standards such as suitability for the task and error tolerance[21]. It also aligns with

Shneiderman's Golden Rules by emphasizing consistency, informative feedback, and easy reversal of actions.[22] Together, these standards help reduce complexity, minimize training requirements, and promote adoption across diverse teams.

3.5.1 System Level Use Case

ReportBuilderPro lets users create, manage, and approve construction reports, and it uses AI to report problems that might not have been found otherwise. It makes sure that the layers for entering, processing, and displaying data work integrate perfectly so that reports are accurate and easy to understand.

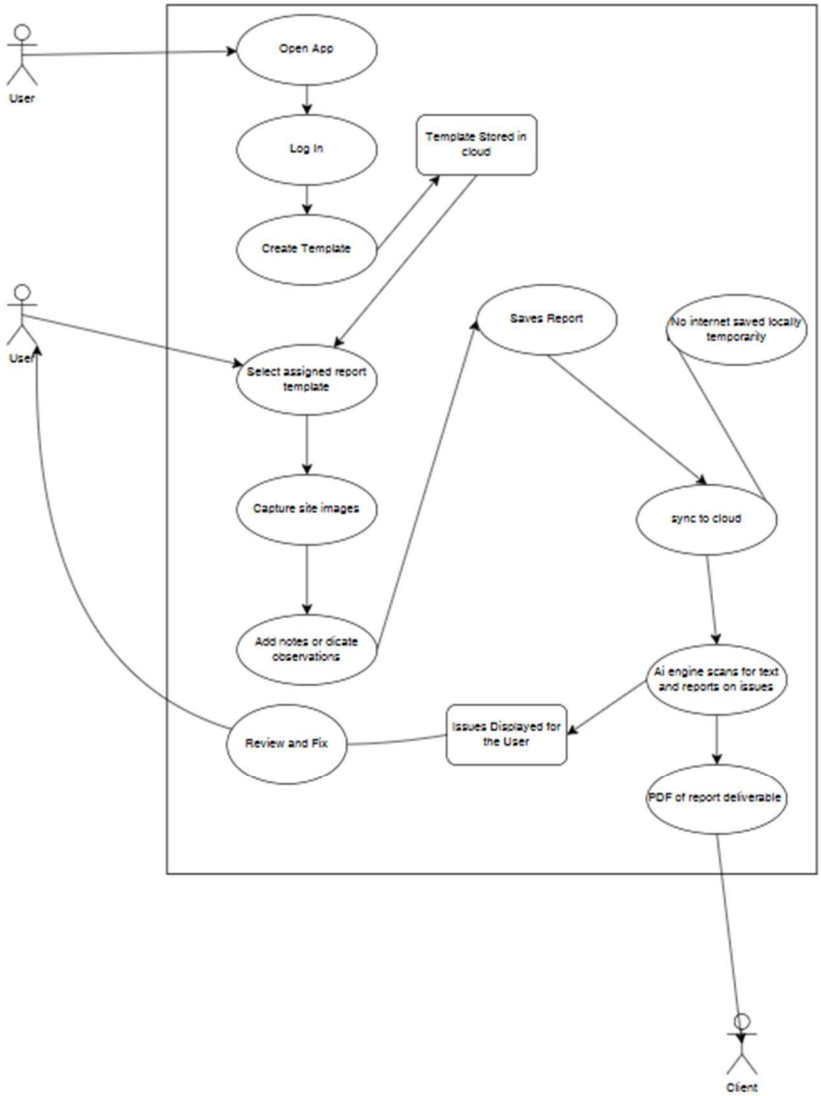


Figure 3 System Level Use Case

3.5.2 User Creates Report Template

This use case makes sure that everything is consistent and follows the rules by letting managers create standard templates that meet both project and regulatory needs.

USE CASE	1	Project Manager Creates Standardised Report Template
Description of Goal in Context	Enable the Project Manager to create a standardized report template for site documentation, ensuring consistency and compliance across projects.	
Preconditions	Manager is authenticated in the web application. Manager has appropriate permissions to create templates. Web application is operational.	
Post Conditions, Success End Condition	Template is saved in the system and available for assignment to projects. Confirmation message displayed to the manager.	
DESCRIPTION of Scenario	The Project Manager logs into the web application, navigates to the template builder, and creates a new report template by adding sections (e.g., progress photos, safety checks, notes). Once finalized, the template is saved and becomes available for use by on-site personnel..	

Main Flow

Step	Action	Response
1	Log into web application	Authentication verified
2	Navigate to Template Builder	Display template creation interface
3	Add sections (images, notes, checklists)	System validates input
4	Finalise template configure any mandatory fields	System applies settings
5	Save template	Confirmation message displayed
6	Template available for use	System updates template list

EXCEPTIONS or ERROR Flow
Description

Step	Branching Action	Alternate
1	App crashes during template creation	Auto-save draft and prompt recovery.

ALTERNATIVE or VARIATION Flow
Description
Variations in how the report is submitted depending on connectivity

Step	Branching Action	Alternate
1	Manager edits an existing template instead of creating a new one.	
2	Manager duplicates an existing template for faster setup.	

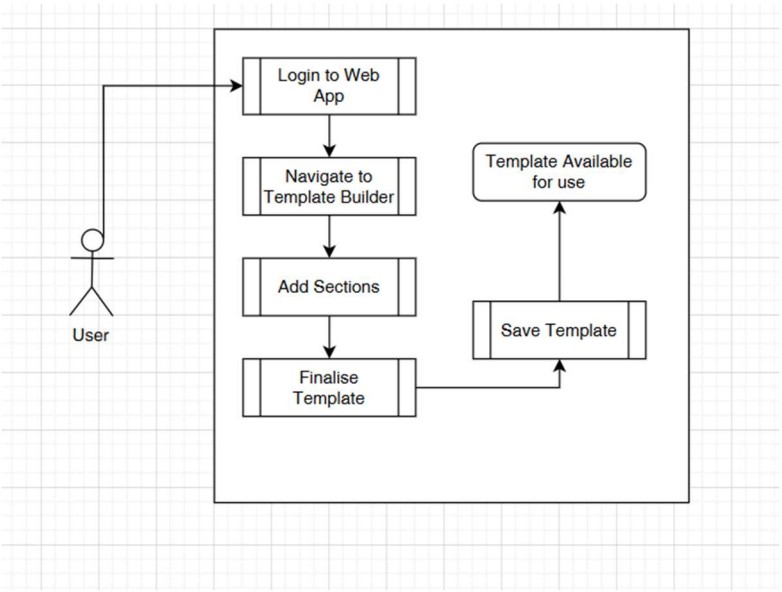


Figure 4 User Creates Report Template

3.5.3 User Fills in Report On-Site

Capturing real time site data through structured templates improves accuracy, reduces reporting delays, and provides a solid foundation to create the finalised report.

USE CASE	2	Project Manager uses predefined report template to document site progress
Description of Goal in Context	Enable the Project Manager to document site progress during a visit by completing a predefined report template, capturing images, and adding observations for later AI analysis and finalisation for client delivery.	
Preconditions	Mobile app installed and authenticated. Assigned report template available. Device has sufficient storage for offline mode.	
Post Conditions, Success End Condition	Report saved locally or synced to cloud. Confirmation message displayed to user.	
DESCRIPTION of Scenario	The Project Manager opens the mobile app on-site, selects the assigned report template, captures progress photos, and adds notes or dictated observations. If offline, the report is stored locally and queued for sync. Once online, the report is uploaded to the cloud for finalisation on web application.	

Main Flow

Step	**Action**	**Response**
1	Open mobile app	Verify authentication
2	Select assigned report template	Display template fields
3	Capture site images	Store locally
4	Add notes or dictate observations	Validate input
5	Save report	Confirm save
6	Sync report to cloud	Show sync progress & success

EXCEPTIONS or ERROR Flow
Description

Step	**Branching Action**	**Alternate**
1	App crashes during report creation	Auto-save draft and prompt recovery.
	Image upload fails	Retry prompt.

ALTERNATIVE or VARIATION Flow
Description
Variations in how the report is submitted depending on connectivity

Step	**Branching Action**	**Alternate**
1	Offline Mode: Report stored locally and queued for sync.	
2	Template Error: Prompt user to reload or select another template.	

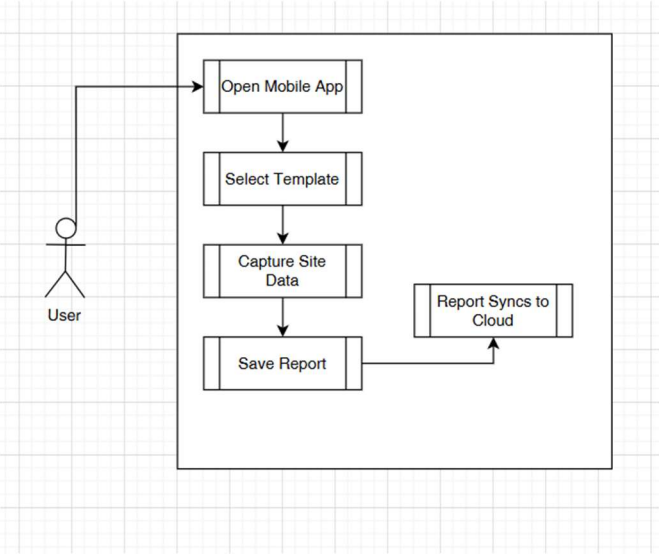


Figure 5 User Fills Report on site

3.5.4 User Completes the Report in the Web Application

Finalising reports in the web app lets you review them in depth. The user can start AI analysis, which helps find risks early and make smart decisions.

USE CASE	3	Project Manager completes Report on Web App and Submits for AI analysis
Description of Goal in Context	Enable the Project Manager to finalize a completed report on the web application and trigger AI analysis for risk detection, delays, and material shortages.	
Preconditions	Report has been created and synced from the mobile app. Manager is authenticated in the web application. AI engine is operational and accessible.	
Post Conditions, Success End Condition	Report finalized and locked for analysis. AI engine processes report and displays flagged issues on dashboard. Confirmation message displayed to user.	
DESCRIPTION of Scenario	The Project Manager logs into the web application, reviews the synced report from the mobile app, makes any necessary edits, and finalizes it. They then trigger AI analysis, which scans the report text and images for risks, delays, and shortages. Results are displayed on the dashboard for review.	

Main Flow

Step	**Action**	**Response**
1	Log into web application	Authentication verified
2	Open synced report	Display report details
3	Review and edit report	Validate changes
4	Finalize report	Lock report for analysis
5	Trigger AI analysis	Show progress indicator
6	Display flagged issues	Update dashboard with results

EXCEPTIONS or ERROR Flow
Description

Step	**Branching Action**	**Alternate**
1	AI engine fails	Display error and allow retry.
	Report validation error	Prompt correction before analysis.

ALTERNATIVE or VARIATION Flow
Description
Variations in how the report is submitted depending on connectivity

Step	**Branching Action**	**Alternate**
1	Manager cancels AI analysis	Report remains editable.
2	AI analysis delayed due to server load	Notify user and queue task.

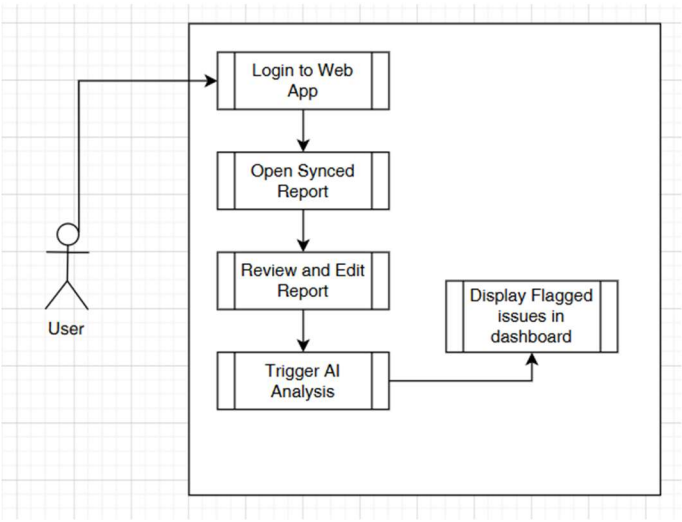


Figure 6 User completes report in web app

3.5.5 Use Case Justification

The chosen use cases were selected because they are the main workflows in construction reporting and directly address the problems that were found during stakeholder analysis and requirements gathering. Each use case fits with the system's goals of making things more efficient, accurate, and compliant.

User Creates Report Template This ensures standardisation and regulatory compliance by allowing managers to design structured templates tailored to project needs. This addresses the lack of consistency and manual template creation highlighted as a major pain point.

User Fills Report On-Site This shows real time site data capture using predefined templates, reducing delays and miscommunication caused by manual notetaking and image compilation. Offline-first capability ensures reliability in low-connectivity environments.

User Completes Report in Web Application enables thorough review and finalisation of reports, incorporating AI-driven analysis for risk detection and material shortage identification. This supports proactive decision making and improves overall reporting quality.

These use cases were given top priority because they are the most important parts of the reporting process i.e. creating templates, capturing data, finalising reports, and analysing them. They all work together to make sure that ReportBuilderPro has a smooth, user-friendly process that solves problems in the industry like broken communication, compliance risks, and slow manual workflows.

3.6 Preliminary High-level Architecture

ReportBuilderPro's architecture will be built to support offline-first reporting, secure data handling, and AI-driven analysis. This will make sure that it works well in construction sites where the internet connection is not always stable.

The system will consist of three main layers frontend, backend, and storage. The frontend will include a web application built with React and a mobile application built with React Native. These will provide user interfaces for creating templates, capturing site data, and managing reports. Authentication will be handled through Azure Active Directory to ensure secure access and role-based permissions.

The mobile app will let you cache data offline, so you can still collect it even if you're not connected to the internet. When the connection is back up, a syncing engine will send data to the backend. FastAPI and Node.js will be used to build the backend, which will handle business logic and work with the AI/NLP engine. The AI engine will use spaCy to clean up text and Scikit-learn to sort things like finding risks, delays, and shortages of materials.

MongoDB will hold structured and semi-structured report content, and AWS S3 will hold images and PDF reports. After the reports are synced, the AI issue scanner will go through them and mark any possible problems. These will be shown on the dashboard for you to look at. After that, reports will be saved as PDFs so that they can be shared and kept in compliance.

This architecture will ensure security, and responsiveness while addressing key challenges such as intermittent connectivity and the need for intelligent reporting.

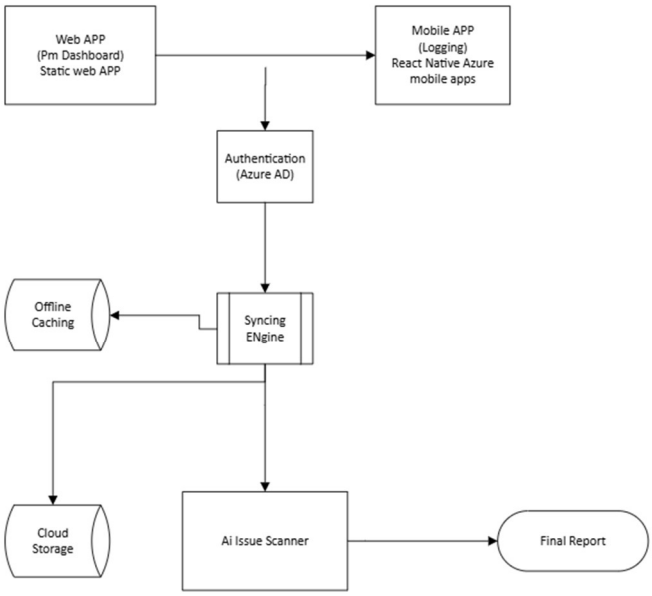


Figure 7. ReportBuilderPro System Workflow

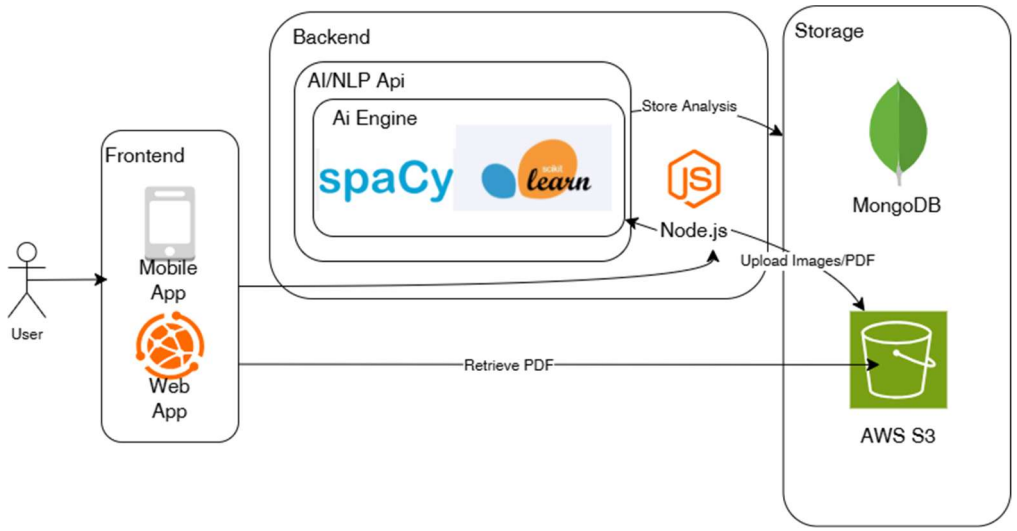


Figure 8 ReportBuilderPro System Architecture

3.7 Key Components Breakdown

This section outlines the main subsystems of ReportBuilderPro, focusing on their roles, the technologies they use, and how they work together to make site reporting and project oversight easy.

1. Web Application

Purpose

The Web application will provide administrative capabilities for ReportBuilderPro and act as the central hub for the application.

Core Features Include

Template Builder

The system will include a fully customisable template builder that will allow users to create report templates tailored to different project types. This feature will promote consistency and adaptability across various use cases.

Project Dashboard

A dynamic project dashboard will provide visual tracking of key metrics, issues, and overall progress. This functionality will facilitate data-driven decision-making and enhance project oversight.

PDF Report Generation

The system will support exportable, standardised PDF reports for easy sharing and compliance with industry documentation practices. This will ensure that reporting remains professional and aligned with regulatory standards.

2. Mobile Application

Purpose: The mobile application will allow users to capture site data in a simplified manner user their pre-defined templates created in the web application even in low connectivity areas.

Core Features Include

Offline First Reporting

The application will provide local data storage with deferred synchronisation, ensuring uninterrupted data capture in remote or signal-poor areas.

Image Capture

It will support multimedia inputs to enhance report richness and context, enabling visual documentation and hands-free reporting.

Backend Synchronisation

The system will automatically synchronise data when connectivity is restored, maintaining data integrity and continuity across devices.

3. Backend & API Layer

Purpose: The backend and API layer will serve as the system's data and logic core, enabling secure storage, processing, and integration across all components.

MongoDB

The system will use a NoSQL database for flexible document storage, which is well-suited for handling unstructured and semi-structured report data.

FastAPI

A lightweight Python-based framework will be implemented for API development, enabling rapid development and seamless integration with AI modules.

AWS (Amazon Web Services)

The solution will leverage cloud-based file storage and processing through AWS, ensuring scalability, reliability, and secure file handling.

4. AI/NLP Feature

The AI/NLP feature will enhance report intelligence by applying automated text analysis and risk detection to improve decision-making and project oversight. This capability will leverage Natural Language Processing techniques to interpret report content, identify patterns, and flag potential issues proactively.

The system will analyse report text to classify information into categories such as delays, risks, and material shortages. Detected issues will be highlighted on the dashboard for user review, and users will have the option to trigger AIanalysis manually after completing a report. This approach ensures flexibility and user control while maintaining transparency in AI driven insights.

To deliver a responsive experience, NLP processing will be designed to complete within three seconds for standard reports, and the AI engine will aim for at least 85% accuracy in classification tasks based on validation datasets. Clear feedback will be provided when analysis fails or is delayed due to server load, and all AI outputs will be advisory rather than authoritative, clearly marked as system-generated insights.

These design choices reflect a balance between intelligence and usability. By focusing on speed, accuracy, and transparency, the AI/NLP feature will address stakeholder needs for proactive risk detection while aligning with industry trends in intelligent document processing.

3.8 User Interface Mock-ups

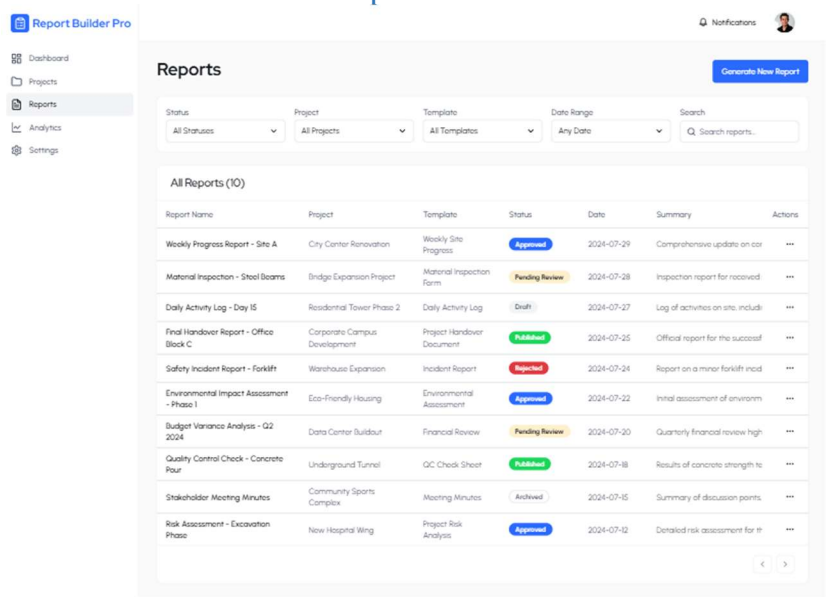


Figure 9. Sample Ui Mock-up 1

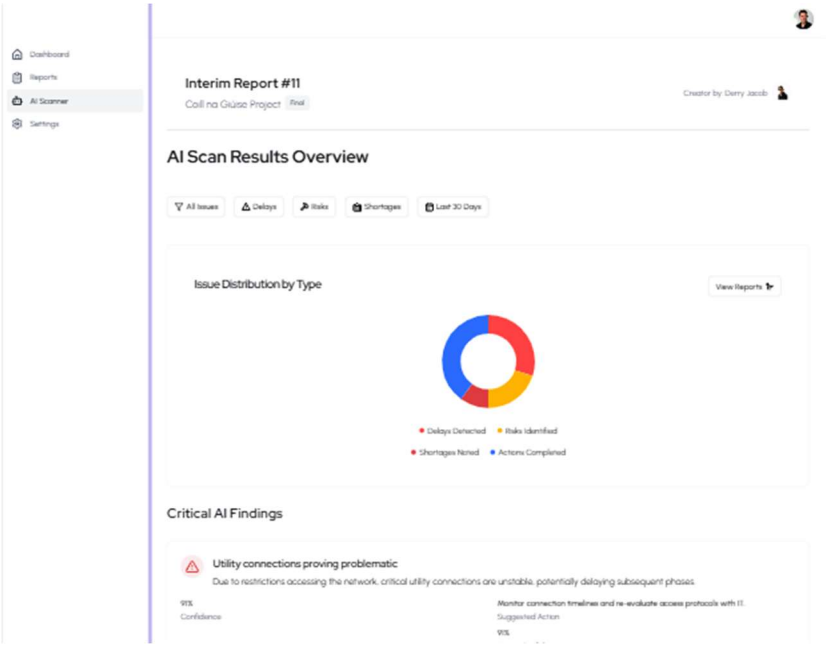


Figure 10.Sample Ui Mock-up 2

3.10 UX and Accessibility Requirements

The UX design of Report Builder Pro is central to its success, particularly given its role in facilitating structured reporting in dynamic, field based environments. The application is designed with a human centred approach, prioritising the needs, behaviours, and constraints of its end users.

To ensure relevance and usability, user research was conducted (see Section 2) Aswell as clearly defined use cases (see Section 3.6), focusing on pain points in current reporting workflows, such as time-consuming data entry, lack of mobile responsiveness, and difficulty accessing summarised insights. In future versions, user testing will be used to see how well the prototype works and to make changes to features based on user feedback.

The design is based on well-known principles of usability and accessibility, which makes sure that Report Builder Pro provides a smooth, quick, and welcoming reporting experience even in tough field conditions.

3.10.1 Shneiderman's Eight Golden Rules of Interface Design [22]

Shneiderman's rules guide the interface design of Report Builder Pro, ensuring that users can report quickly, accurately, and confidently in the field.

Strive for Consistency
Report templates, navigation patterns, and UI components are consistent across mobile and desktop, reducing training time and supporting cross-platform use on-site and in the office.

Enable Universal Usability
The platform supports a range of devices and user capabilities, including rugged tablets and low-bandwidth environments, ensuring accessibility for all field roles.

Offer Informative Feedback
Real-time feedback (e.g., sync status, submission confirmations) helps users understand system responses, critical for time-sensitive reporting.

Design Dialogs to Yield Closure
Reporting workflows are structured to guide users to completion, with clear summaries and confirmations that reinforce task success.

Prevent Errors
Smart form validation and contextual prompts reduce the risk of incorrect data entry, which is especially important when reporting under pressure or in poor conditions.

Permit Easy Reversal of Actions
Users can edit or undo entries, reset filters, and revise reports—supporting flexibility and reducing anxiety around mistakes.

Support Internal Locus of Control
Users initiate and control reporting actions, from choosing templates to customizing summaries, reinforcing confidence and autonomy.

Reduce Short-Term Memory Load
Frequently used tags, categories, and templates are surfaced contextually, allowing users to focus on the task rather than remembering system details.

3.10.2. ISO Dialogue Principles [21]

Report Builder Pro also aligns with the ISO 9241-110 standard for dialogue principles, ensuring usability and accessibility in professional environments

Suitability for the Task
Interfaces are tailored to construction reporting tasks, using industry-specific terminology and workflows (e.g., “Safety Issue,” “Site Condition”).

Self-Descriptiveness
Each screen and input field clearly communicates its function, helping users understand what to do without needing external guidance.

Controllability
Users can navigate freely, pause and resume reporting, and control data inputs critical for field users who may be interrupted or multitasking.

Conformity with User Expectations
The design reflects familiar patterns from construction and safety reporting tools, reducing friction and improving adoption.

Error Tolerance
The system supports recovery from mistakes with clear error messages and suggestions (e.g., “Please enter a valid location”), minimizing disruption.

Suitability for Individualisation
Users can adjust layout, font size, and themes to suit their working conditions, such as bright outdoor environments or low-light indoor settings.

Suitability for Learning
Embedded guidance, tooltips, and onboarding flows help new users get started quickly, while experienced users can skip or customize their experience.

3.10.3. Accessibility and ARC Toolkit Integration

To ensure inclusivity and compliance with modern standards, Report Builder Pro integrates accessibility testing using the ARC Toolkit, and follows WCAG 2.2 [23] guidelines.

Colour Contrast Testing

The UI colour palette is tested to meet contrast ratios for readability, especially in outdoor lighting conditions.

Keyboard Navigation

All features are accessible via keyboard, supporting users with motor impairments and improving accessibility on rugged field devices.

Customizable UI

Users can adjust text size, switch to high-contrast themes, and enable simplified layouts for better visibility and focus.

Screen Reader Compatibility

Semantic HTML and ARIA labels are used to ensure compatibility with assistive technologies.

Touch-Friendly Design

Buttons and input fields are sized appropriately for touch interaction, reducing errors and improving usability on mobile devices.

3.11 Error message and Feedback Design

ReportBuilderPro will prioritize clear, actionable feedback such as:

Error Messages

Tailored messages like “Please enter a valid format image i.e. png or jpg” will guide users to correct inputs.

Success Feedback

The creation of subtle animations and checkmarks will confirm completed actions, such as saving a pdf or AIanalysis completion.

Responsiveness

Smooth transitions and quick load times will reinforce a sense of system reliability and efficiency to users.

3.12 Conclusions

This chapter outlined the system analysis for ReportBuilderPro, covering everything from stakeholder requirements and functional specifications to system architecture, UI mock-ups, and technology choices. It established a clear, user-centred foundation for development, guided by Feature-Driven Development, accessibility standards, and proven UX principles like Schneiderman’s Golden Rules[22] and ISO 9241-110 [21]. Together, these elements ensure the platform is intuitive, scalable, and purpose-built for efficient, cross-device reporting in demanding field environments.

4. System Design

4.1 Introduction

This section introduces the approach taken to transform the requirements from Section 3 (System Analysis) into a structured and system design for ReportBuilderPro. The goal is to make sure that the construction industry's system architecture, parts, and workflows meet the real needs of field workers and project stakeholders.

The design process is guided by a combination of:

- **Feature-Driven Development (FDD)** to support iterative delivery and continuous feedback.
- **User-centred design principles** to ensure usability, accessibility, and relevance in real-world site conditions.
- **Modular architecture** to enable flexibility, maintainability, and future scalability.
- **Best practices in cloud infrastructure and AI integration**, ensuring the system is robust, secure, and capable of intelligent reporting.

4.2 Software Development Methodology

To guide the development of ReportBuilderPro, a Feature-Driven Development (FDD) approach was adopted. FDD is a client-centric, iterative methodology that focuses on designing and building software around clearly defined, user-valued features. This approach aligns strongly with the goals of ReportBuilderPro, which prioritises usability, modularity, and responsiveness to real-world site reporting needs.

As outlined by Scott W. Ambler in *"Feature Driven Development (FDD) and Agile Modeling"*[15], FDD is built on five core activities:

Develop an Overall Model

- Core entities such as users, reports, templates, risk flags, and workflows were mapped out.
- This model was informed by expert elicitation using the IDEA Protocol [14] (see Section 2.4.1).

Build a Feature List

- A comprehensive list of desired features was compiled based on stakeholder input and workflow analysis.

Plan by Feature

- Features were prioritised based on user feedback, feasibility, and impact.
- Logical grouping into modules (e.g., reporting, dashboard, risk analysis) supported structured planning.

Design by Feature

- Each feature was designed with a focus on simplicity, accessibility, and relevance to field workflows.
- UX principles such as Shneiderman's Golden Rules [22] and ISO 9241-110 [21] were applied to ensure intuitive interaction.

Build by Feature

- The system architecture supports modular, feature-based development, enabling iterative delivery and continuous refinement.

FDD Principles Applied to ReportBuilderPro

Feature-Centric
Each feature represents a small, client-recognisable unit of functionality (e.g., "Submit Site Report," "Flag Risk," "Generate PDF").

Iterative and Incremental
Features are developed in short cycles, allowing for ongoing feedback and adaptation to evolving site needs.

Domain-Driven
The system is modelled around real-world entities and processes in construction reporting, ensuring relevance and clarity.

Team-Oriented
Roles such as Feature Owner and Class Owner (as described by Ambler [15]) promote accountability and collaboration across design and development.

As Ambler states, *"FDD blends several industry-recognized best practices into a cohesive whole. These practices are driven from a client-valued functionality (feature) perspective."* [15] This makes FDD particularly well-suited to ReportBuilderPro, where stakeholder engagement and field usability are central to success.

4.2.1 Comparison between FDD and other methodologies

While Agile and Scrum are also widely adopted methodologies in software development, Feature-Driven Development (FDD) was selected for ReportBuilderPro due to its stronger alignment with the project's structure and goals.

Comparative Analysis of Development Methodologies

Methodology	Characteristics	Relevance to ReportBuilderPro
FDD	- Feature-centric - Structured around domain modelling and feature lists - Focuses on designing and building by feature	FDD provides a clear path from requirements to implementation, making it ideal for a system like ReportBuilderPro that is built around user-recognisable reporting features.
Agile [24]	- Values individuals, collaboration, and adaptability - Emphasizes working software and customer feedback	Agile principles underpin the overall development philosophy, especially in terms of iterative progress and responsiveness to user needs.
Scrum[25]	- Framework within Agile - Uses sprints, roles (Scrum Master, Product Owner),	Scrum offers strong team coordination, but its sprint-based structure may be less

	and ceremonies (stand-ups, retrospectives)	suited to the modular, feature-first nature of this project.

Decision Matrix for Development Methodology – ReportBuilderPro

A decision matrix was made to help choose the best software methodology for ReportBuilderPro. The criteria used were directly related to ReportBuilderPro's goals and limits. Some of these criteria were user-centred design, modular feature delivery, support for domain modelling, iterative development, stakeholder engagement, and scalability for complex systems.

Every method, such as We looked at FDD, Agile, and Scrum based on these factors and how well they fit with the basic ideas of a feature-first, field-oriented reporting platform. Feature-Driven Development (FDD) was the best choice because it has a structured way of building features that clients can recognize, it focuses on domain modelling, and it works well with modular system architecture. While Agile and Scrum offer valuable practices, FDD provides the clearest path from stakeholder requirements to implementable features, making it the best fit for ReportBuilderPro's development strategy.

Criteria	FDD	Agile	Scrum
User-Centric Design	Yes	Yes	Yes
Modular Feature Delivery	Yes	Partial	Partial
Domain Modelling Support	Yes	No	No
Iterative Development	Yes	Yes	Yes
Team Roles & Accountability	Yes	Yes	Yes
Stakeholder Engagement	Yes	Yes	Yes
Suitability for ReportBuilderPro	Yes	No	No

4.3 High Level System Architecture

This section shows the high-level architecture and infrastructure of ReportBuilderPro, integrating insights from earlier analysis and design decisions. It outlines the logical structure of the system, the initial physical infrastructure supporting development and testing, and considerations for deployment in a production environment.

4.3.1 Logical Architecture

The logical architecture of ReportBuilderPro will be modular and feature-driven, reflecting the principles outlined in Section 4.2 (Software Methodology) and the component breakdown in Section 3.5. Will be composed of four interconnected subsystems.

1. Web Application

The web application will serve as the central hub of ReportBuilderPro, providing a platform

for administrative tasks, template creation, and report refinement. It will enable users to create and customise report templates, display flagged risks, delays, and material shortages through an interactive dashboard, and support PDF generation for client-facing documentation. This component will act as the primary interface for project managers and admin ensuring comprehensive oversight and control.

2. Mobile Application
The mobile application will be developed to facilitate efficient on-site data capture, particularly in environments with limited connectivity. It will operate on an offline-first basis, storing data locally and synchronising automatically when a connection becomes available. Users will be able to upload images, dictate notes, and complete structured checklists directly from the field. This functionality will be tailored for operatives such as project managers and quantity surveyors, ensuring accurate and timely reporting from remote locations.

3. Backend & API Layer
The backend will function as the system's core for data processing and communication, ensuring secure and reliable operations. It will manage user authentication and role-based access control, handle data storage, file uploads, and report submissions, and provide API endpoints to facilitate seamless communication between the mobile and web applications. This layer will underpin the system's scalability and maintain data integrity across all components.

4. AI/NLP Engine
An integrated AI engine will enhance the intelligence of ReportBuilderPro by analysing report content using advanced Natural Language Processing techniques. It will scan text for risks, delays, and material shortages, apply machine learning models to classify and flag issues, and continuously improve through feedback and usage patterns. This subsystem will provide proactive insights to support decision-making while maintaining transparency and user control.

4.3.2 Physical Infrastructure
The infrastructure will be cloud-based to support rapid development, secure testing, and modular deployment.

Hosting: The web application and backend services will be hosted on Azure, selected for its integration with development tools and scalability.

Storage: Uploaded images and generated PDFs will be stored securely using AWS S3. MongoDB Atlas will be used for flexible, document-oriented data storage.

Networking: All data will be transmitted over HTTPS [26], with secure access protocols in place to protect user information.

4.3.3 Deployment and Production Environment
Although full-scale commercial deployment is outside the scope of the current phase, the system will be designed with production-readiness in mind.

Environment Setup Separate development, testing, and production environments will be established to support iterative development and quality assurance.

Deployment Strategy CI/CD pipelines will be implemented to automate deployment and reduce manual errors.

Security and Compliance: The system will comply with GDPR standards and include encryption for data at rest and in transit.

Monitoring and Maintenance Logging and alerting mechanisms will be used to monitor system health and performance, with modular architecture supporting future scalability.

4.4 Component Level Design

The design of ReportBuilderPro is guided by the principles of Feature-Driven Development (FDD), a methodology that emphasizes building software around clearly defined, user valued features. This approach ensures that the system is modular and aligned with real-world workflows in the construction industry

4.4.1 Development Methodology Design Considerations

FDD is structured around five key activities:

1. **Develop an Overall Model** – Define core entities and relationships.
2. **Build a Feature List** – Identify user-valued features based on stakeholder input.
3. **Plan by Feature** – Prioritize and group features into logical modules.
4. **Design by Feature** – Create design specifications for each feature.
5. **Build by Feature** – Implement features iteratively and incrementally.

This methodology was selected for its alignment with the modular architecture of ReportBuilderPro and its ability to support continuous refinement based on user feedback.

4.4.2 Module Design

ReportBuilderPro will be structured around five core modules, each designed to deliver a specific set of prioritised features that collectively enable a robust and user-friendly reporting solution.

1. Reporting Module

The Reporting Module will form the foundation of ReportBuilderPro's functionality, enabling users to create and customise report templates, capture structured data on-site via the mobile application, and submit completed reports for AIanalysis. Templates will be developed using reusable UI components in React to ensure consistency and maintainability. Mobile forms will support offline-first data entry, allowing operatives to record information even in low-connectivity environments. All reports will be stored in MongoDB, leveraging its flexible schema to accommodate diverse reporting requirements.

2. Dashboard & Analytics Module

The Dashboard & Analytics Module will act as the central interface for supervisors and decision-makers within ReportBuilderPro. It will visualise flagged risks, delays, and material shortages, while presenting AI-generated insights and recommendations in an intuitive format. Dynamic data binding and filtering will ensure real-time updates, and AI outputs will be integrated through FastAPI endpoints. Visual cues and icons will be incorporated to support rapid interpretation and informed decision making.

3. AI/NLP Module

Features

The AI/NLP Module will provide the intelligence layer for ReportBuilderPro, analysing report content to identify risks, delays, and material shortages. It will classify issues using trained machine learning models and continuously improve through feedback loops. The NLP pipeline will utilise spaCy for preprocessing, while classification models will be implemented using Scikit-learn. This module will transform raw data into actionable insights, reinforcing ReportBuilderPro's role as a proactive reporting solution.

4. PDF Generation Module

The PDF Generation Module will enable users to export reports in a professional, standardised format that includes AI insights and visual summaries. HTML templates will be converted into PDFs using pdfkit, and layouts will be optimised for readability and compliance with industry documentation standards. This feature will ensure that ReportBuilderPro outputs meet client and regulatory expectations.

5. Authentication & Access Control Module

The Authentication & Access Control Module will secure ReportBuilderPro by providing robust login functionality and enforcing role-based access permissions across mobile and web platforms. Authentication will be managed through the Node.js backend, and access rules will be applied at both API and UI levels to maintain system integrity and protect sensitive project data.

The design system will follow key principles to ensure usability.

- **Modularity** Each feature will be encapsulated in its own component or service.
- **Accessibility** UI components will be tested using ARC Toolkit and follow WCAG 2.2 guidelines.[22]
- **Consistency** Design patterns will be reused across mobile and web interfaces.
- **Error Tolerance** Smart validation and feedback mechanisms will reduce user errors.

4.4.3 Visual Design

The visual design will be guided by:

Material Design principles for layout and interaction.

Responsive design to ensure usability across devices.

Touch friendly components for mobile field use.

High contrast themes for outdoor visibility.

4.5 Data Design

To support the modular and feature-driven architecture of ReportBuilderPro, a conceptual Entity Relationship Diagram (ERD)[27] has been developed. This ERD outlines the core entities within the system and the relationships between them, ensuring that the data model aligns with the reporting workflows and AI-enhanced analysis features described in earlier sections.

The ERD is designed to reflect the following principles:

- **Modularity** Each entity corresponds to a distinct feature or subsystem.
- **Clarity** Relationships are kept intuitive to support maintainability and ease of development.

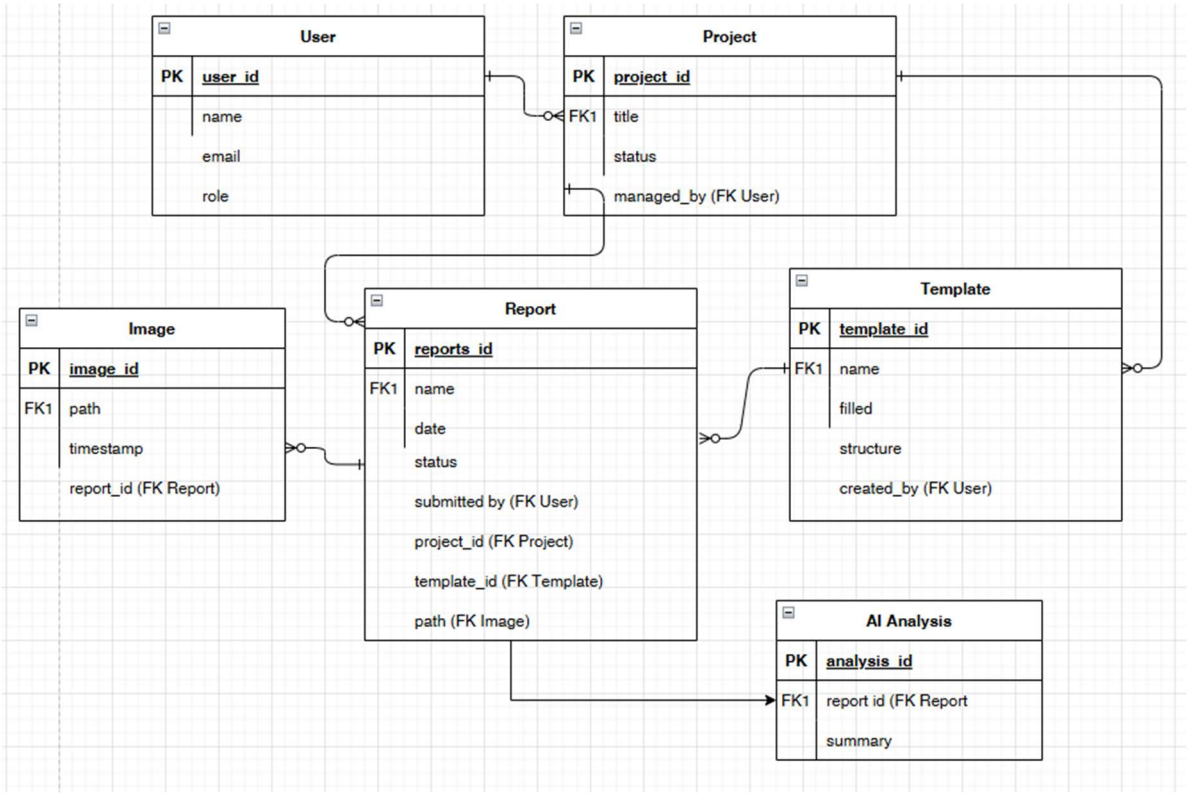


Figure 11 ReportBuilderPro Conceptual ERD

Document Data Model (MongoDB)

ReportBuilderPro will store reports and other data in MongoDB in a way that is based on documents. An example structure:

```
{
  "report_id": "R123",
  "project_id": "Wex1234",
  "submitted_by": "Derry",
  "date": "2025-11-20",
  "status": "completed",
  "template": {
    "template_id": "TMP001",
    "sections": ["progress", "health_safety", "risks"]
  },
  "images": [
    {
      "image_id": "IMG001",
      "path": "s3://reportbuilderpro/images/img001.jpg",
      "timestamp": "2025-11-20T10:30:00Z"
    }
  ],
  "ai_analysis": {
    "analysis_id": "AI001",
    "summary": "Risk detected: material shortage",
    "flags": ["material_shortage"]
  }
}
```

Figure 12 Sample of data model

Data Validation Rules

To keep data safe and make sure that all reporting workflows in ReportBuilderPro are processed the same way, strict validation rules will be used. To make sure that each report is complete, it must have required fields like report_id, project_id, submitted_by, date, and

status. The status attribute will only be able to have the values "draft," "submitted," or "completed," which will make it clear what stage the workflow is in. To stay in line with cloud storage standards, image paths will follow the AWS S3 URL format. AI analysis flags will be checked against pre-set groups like risk, delay, and material shortage to make sure they are classified and reported correctly

4.6 Feature to Component mapping

The table below demonstrates how each feature in ReportBuilderPro is supported by a specific system component, reinforcing the modular design [28] and ensuring traceability between functional requirements and technical implementation.

Feature	System Component	Technology	Design Note
Submit Site Report	Mobile application	React	Offline first design with local caching and deferred sync
Create and Edit Report template	Web Application	React.js	Drap and drop interface for template customisation
Upload Images and notes	Mobile Application	React Native, AWS S3	Support multimedia inputs and sync when online
AI Scan	NLP Engine	spaCy, Scikit-learn, FastAPI	NLP pipeline for text analysis and classification of issues
Generate PDF Report	Backend	Pdfkit	Convert structure format to professional PDF
View Dashboard and Insights	Web Application	React.js, FastAPI	Display flagged issues in real time
User Login and Access Control	Backend	Node.js, MongoDB	Role based secure authentication
Store and Retrieve Data	Backend	MongoDB	Flexible schema to support varied reports
Sync Mobile and Web Data	Backend	FastAPI , Node.js	Ensure seamless data flow between mobile and web

4.7 Outline of ReportBuilderPro Tech Stack

ReportBuilderPro Tech Stack

Frontend	React/React Native
Preprocessing	spaCy
Model	Scikit-learn
Api	FastAPI
Database	MongoDB
Pdf export	pdfkit
Hosting	AWS
NLP Processing	spaCy
ML Models	Scikit-learn
Accessibility Testing	ARC Toolkit

4.8 Natural Language Processing (NLP) Design

Natural Language Processing will be a big part of ReportBuilderPro's smart reporting features. Unlike traditional reporting systems like SiteCam and Fieldwire, ReportBuilderPro will use NLP to automatically understand unstructured text and find useful information. These reports from users point out risks, delays, and material shortages right away.

Adding NLP is not just a backend improvement, it's a core feature that sets ReportBuilderPro apart and supports many user-facing features. NLP is an important part of the system's architecture and fits with the Feature-Driven Development (FDD) approach used for this project. It can flag problems on the dashboard and make AI-enhanced summaries for PDF exports.

Recent literature emphasizes that NLP systems must be designed not only for performance, but also with ethical integrity in mind. Leidner and Plachouras (2017) [23] argue for an "ethical by design" approach, where ethical considerations are embedded throughout the entire lifecycle of NLP development, from data collection and model training to deployment and user interaction. This is especially important for ReportBuilderPro, which works with private site data and makes automated insights. The system's NLP engine will be built using clear methods, trained on data specific to the field, and tested not only for accuracy but also for fairness, privacy, and user impact, in order to follow these rules.

Design Overview

The NLP engine will be structured as a modular pipeline, enabling flexibility, maintainability, and continuous improvement.

Preprocessing

- Tokenization, sentence segmentation, and entity recognition using **spaCy**.
- Cleaning and normalization of report text to reduce noise and improve model input quality.

Classification

- Use of Scikit-learn to train supervised models that classify report content into categories such as:

- Risk
- Delay
- Material Shortage
- Models will be trained on annotated construction report data and refined through iterative feedback.

Integration

- The NLP engine will be exposed via **FastAPI**, allowing seamless communication with the dashboard and backend.
- Flagged issues will be returned as structured data for display in the dashboard and inclusion in PDF summaries.

Output:

- Each report will be enhanced with a list of flagged items, confidence scores, and suggested actions

NLP Pipeline Overview

Stage	**Description**	**Technology/Method**
Text Access	Retrieve report text from the database or file storage.	MongoDB, AWS S3
Preprocessing	Clean and prepare text for analysis: tokenization, lemmatization, etc.	spaCy
Feature Extraction	Convert text into numerical features for model input	Scikit-learn, spaCy
Classification	Apply trained models to classify content into categories	Scikit-learn
Flag Generation	Generate structured flags with labels and confidence scores.	Custom Logic
Dashboard Display	Send flagged issues to the frontend for visualization and user review.	FastAPI, React.js
Feedback Loop	Capture user feedback on flagged items to improve model accuracy over time.	Backend logging
PDF Integration	Embed NLP insights into the final report output.	pdfkit

4.9 User Interface Design

The user interface (UI) will be designed to provide a clear, intuitive, and efficient experience for all users, ensuring that tasks such as reporting, reviewing, and managing data will be completed with minimal effort and error. The design approach will prioritise usability, accessibility, and responsiveness across devices.

Design Goals

Clarity and Consistency: The system will maintain uniform layouts, terminology, and visual hierarchy across all screens.

Error Prevention and Recovery: Real-time validation, mandatory field indicators, and clear error messages will be incorporated.

Reduced Cognitive Load: Logical grouping of fields, progressive disclosure for advanced options, and familiar interaction patterns will be applied.

Accessibility: The interface will comply with WCAG standards for colour contrast, keyboard navigation, and screen reader compatibility.

Responsiveness: Layouts will be optimised for desktop, tablet, and mobile devices.

Key Components

1. **Navigation**

   ReportBuilderPro will include a persistent top-level navigation menu for quick access to core modules such as Dashboard, Reports, and Settings, supported by search and filter options for efficient data retrieval.

2. **Reporting Forms**

   Reporting forms will maintain consistency through standardised field labels and input types, while inline validation will prevent errors. Users will be able to edit drafts, confirm submissions, and undo actions, with clear success and error messages displayed prominently.

3. **Dashboards**

   Dashboards will present visual summaries of key metrics using charts and tables, with interactive filters for customised views. Export options will allow users to generate PDF and Excel reports for sharing and compliance.

4. **Feedback and Notifications**

   Real time status updates will inform users of submitted reports, alerts will highlight incomplete or overdue tasks, and non-intrusive notifications will communicate system updates effectively.

To make sure that the user interface is clear, efficient, and easy to use on all platforms, it will follow established usability principles and ISO dialogue guidelines. To follow Shneiderman's rule of reducing short-term memory load, the most used report fields and template components will be at the top of the interface, with auto-complete suggestions to help. Shneiderman's rule for informative feedback will help you use color-coded alerts and status messages on dashboards to show system states, warn of risks, and give timely updates. The ISO principle of suitability for the task will be used in mobile design by putting large, touch-friendly controls first for operators who work outside and need to get things done quickly. All

these design choices will work together to make an interface that is easy to use, responsive, and meets the needs of its users.

4.10 Conclusions

ReportBuilderPro's system design is a clear and complete plan for creating a structured, smart reporting platform that meets the needs of construction professionals. The design is based on a modular architecture and follows Feature Driven Development (FDD). This makes sure that each part directly supports user-valued features, from capturing data on mobile devices to using AI to improve analysis.

By aligning the design with real workflows and usability standards such as Shneiderman's Golden Rules and ISO 9241-110, the system is positioned to deliver a seamless and accessible experience across devices and user roles. The integration of an NLP not only differentiates ReportBuilderPro from existing tools like SiteCam and Fieldwire but also empowers users with initiative-taking insights and intelligent reporting capabilities.

The design decisions made in this chapter lay the foundation for a user-centric system. These decisions will shape the development and testing phases, making sure that the final product still meets the needs of all stakeholders and follows best practices in the industry.

5. Testing and Evaluation

5.1 Introduction

This section describes the planned testing strategy and evaluation criteria that will be used in the next stages of development, since the prototype is still in its early stages.

A structured testing and evaluation process is necessary to make sure that Report Builder Pro works well and is reliable. This process tests to ensure that the system works as planned in all possible paths of execution [29] , meets user needs, and follows the design rules that were set during development.

This chapter outlines how Report Builder Pro will be tested and evaluated to provide a clear overview of the system's performance and usability. Testing and evaluation will ensure that Report Builder Pro delivers accurate reporting, intuitive user interactions, and reliable performance across all environments. The approach will be based on the system's logical architecture, which includes

Data Layer Handles input validation, storage, and retrieval of reporting data.

Application Layer Manages report generation, AI summarisation, and workflow logic.

Presentation Layer Provides dashboards, reporting forms, and mobile interfaces.

Each layer will undergo targeted testing to confirm functionality, usability, and compliance with design principles outlined in Section 3 and Section 4.

5.2 Plan for Testing

5.2.1 Reporting Workflow Testing

Objective: Verify that users can create, edit, and submit reports without errors.

Approach: Simulate real-world reporting tasks using predefined templates.

Tests:

- Mandatory field validation.
- Auto-complete suggestions for frequently used fields (Shneiderman's memory load principle).
- Error handling for incomplete or invalid data.

5.2.2 Dashboard Testing

Objective: Ensure dashboards provide accurate, real-time feedback.
Approach: Test colour-coded alerts, status messages, and risk indicators.
Tests:

- Correct display of alerts for overdue reports.
- Real-time updates when data changes.
- Responsiveness across desktop and mobile

5.2.3 Mobile Interface

- **Objective:** Validate usability in outdoor, time-sensitive environments.
- **Approach:** Field testing with touch-friendly controls and simplified workflows.
- **Tests:**

- Button size and spacing for touch accuracy.
- Offline mode behaviour.
- Performance under limited connectivity

Overall Test Plan

Component	Scenario	Expected Result
Reporting Form	Submit with missing mandatory fields	Inline error message displayed
Dashboard Alerts	Overdue report detected	Colour-coded alert shown immediately
Mobile UI	Operator creates report outdoors	Smooth navigation, no input errors

5.3 Plan for Evaluation

Evaluation will focus on usability, efficiency, and compliance with ISO and Shneiderman principles.

Usability Testing: Observe users completing reporting tasks; measure time, error rate, and satisfaction.

Heuristic Review: Apply Shneiderman's rules (e.g., informative feedback, memory load reduction).

Accessibility Audit: Check WCAG compliance for colour contrast, keyboard navigation, and screen reader support.

Performance Metrics: Measure report generation speed and dashboard refresh rates under typical and peak loads.

5.4 Conclusions

Testing and evaluation will confirm that Report Builder Pro meets its functional and usability objectives. By aligning tests with the logical architecture and design principles, the system will deliver accurate reporting, clear feedback, and mobile-friendly workflows. We will use what we learn from evaluations to make small changes that will keep the platform strong and focused on the user.

6. System Prototype

6.1 Introduction

This chapter gives an overview of the first version of ReportBuilderPro, which focuses on setting up basic features like user authentication and a simple dashboard interface. The prototype shows that it can connect to the MongoDB database and includes a temporary library for handling PDFs. This sets the stage for future versions.

6.2 Prototype Development

The main goal of the first prototype was to set up the web application's main hub, which would include the basic structure for navigation, dashboard elements, and template creation placeholders that would be used to build new features in the future.

Login Screen A simple authentication interface allowing users to log in to ReportBuilderPro.

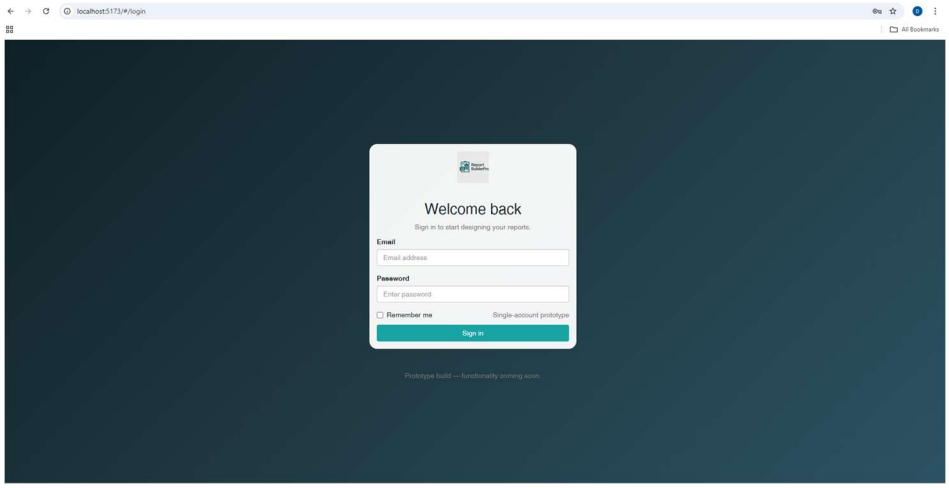


Figure 13 ReportBuilderPro Login Screen

The login screen is a simple way for users to log in, and it connects to a MongoDB database. This integration makes sure that user credentials are stored and retrieved safely.

Security Note Hashing [30] was not implemented in the prototype but will be included in the final version.

```
_id: ObjectId('69222e693fd3fd59a43126b7')
email : "test@hwhpm.ie"
password : "password"
name : "User Test"
```

Figure 14 User Entity in DB

```
// User Login
app.post('/api/login', async (req, res) => {
  const { email, password } = req.body;

  // Find user in MongoDB 'Login' collection
  const user = await db.collection('Login').findOne({ email });

  // Check if user exists and password matches
  if (!user || user.password !== password) {
    return res.status(401).json({ message: 'Invalid email or password' });
  }

  // Success
  return res.json({ email: user.email, name: user.name });
});
```

Figure 15 Prototype Login Route

This code shows how the prototype's basic authentication works. It takes the user's credentials, checks the MongoDB Login collection, and makes sure the email and password are correct. For now, passwords are kept in plain text to make things easier. In the final version, input validation through hashing and safe session handling will be added.

Home Screen Displays a hard coded dashboard and a placeholder for the template creator.

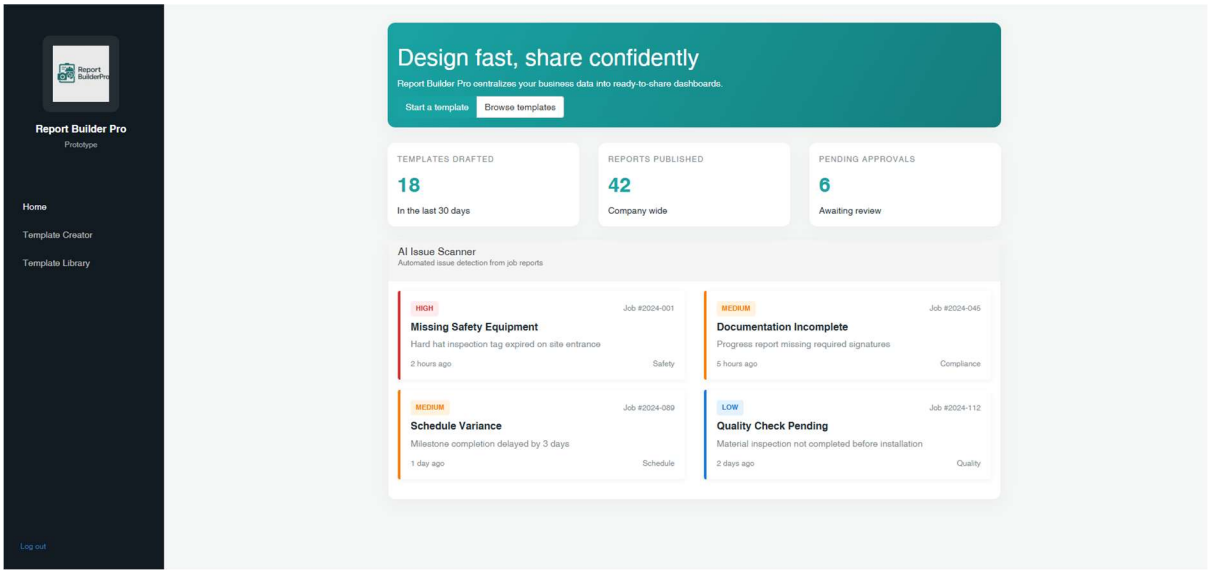


Figure 16 ReportBuilderPro Home screen

```
{/* Hero section with main call-to-action */}
<section className="rbp-hero panel panel-default">
  <div className="panel-body">
    <h1>Design fast, share confidently</h1>
    <p>Report Builder Pro centralizes your business data into ready-to-share dashboards.</p>
    <div className="btn-group">
      <NavLink to="/template-creator" className="btn btn-rbp">
        Start a template
      </NavLink>
      <NavLink to="/template-library" className="btn btn-default">
        Browse templates
      </NavLink>
    </div>
  </div>
</section>
```

Figure 17 Hero Section

The Hero Section is the main call to action on the home page. It tells you what the platform is for and gives you quick ways to make or look through templates. It shows how to use NavLink in React Router to make navigation easy.

When you click on "Start a template" or "Browse templates," you'll go to the right section.

```
{/* Statistics cards */}
<section className="row">
  {stats.map((stat) => (
    <div key={stat.label} className="col-sm-4">
      <div className="panel panel-stat">
        <div className="panel-body">
          <span className="stat-label">{stat.label}</span>
          <strong className="stat-value">{stat.value}</strong>
          <span className="stat-detail">{stat.detail}</span>
        </div>
      </div>
    </div>
  ))}
</section>
```

```
const stats = [
  { label: 'Templates drafted', value: '18', detail: 'In the last 30 days' },
  { label: 'Reports published', value: '42', detail: 'Company wide' },
  { label: 'Pending approvals', value: '6', detail: 'Awaiting review' },
];
```

Figure 18 Statistics Cards and the Sample Data

This code uses React's map() function to dynamically show key performance indicators. Right now, the data is hard coded so that it can be used for demonstration purposes. In the final version, these numbers will come from a backend API that connects to the MongoDB database. This will make sure that they are always up to date based on what users do and system analytics.

```
{/* AI Issue Scanner Section */}
<section className="panel panel-default">
  <div className="panel-heading">
    <h2 className="panel-title">AI Issue Scanner</h2>
  </div>
  <div className="panel-body">
    <div className="row">
      {issues.map((issue) => (
        <div key={issue.id} className="col-sm-6">
          <div className={`issue-card issue-${issue.severity}`}>
            <div className="issue-header">
              <span className={`issue-badge issue-badge-${issue.severity}`}>
                {issue.severity}
              </span>
              <span className="issue-job">Job #{issue.jobId}</span>
            </div>
            <h3 className="issue-title">{issue.title}</h3>
            <p className="issue-description">{issue.description}</p>
            <div className="issue-meta">
              <span className="issue-date">{issue.date}</span>
              <span className="issue-category">{issue.category}</span>
            </div>
          </div>
        </div>
      ))}
    </div>
```

```
const issues = [
  {
    id: 1,
    jobId: '2024-001',
    title: 'Missing Safety Equipment',
    description: 'Hard hat inspection tag expired on site entrance',
    severity: 'high',
    category: 'Safety',
    date: '2 hours ago',
  },
```

Figure 19 AI Issue Scanner Dashboard and Data

These snippets show how Reacts map() function can be used to dynamically show AI-detected problems. For now, the issues are hardcoded to show how to style based on layout and severity.

The AI Issue Scanner will use NLP to look at site reports and inspection documents in the final version. The NLP engine will look through unstructured text for possible safety and compliance problems, sort them by severity and type, and save them in MongoDB. Then, the frontend will get these problems through a secure API, which will let AI-driven alerts show up in real time.

Template Creator

The Template Creator lets you make your own report templates by dragging and dropping them. Users can choose from the left-hand panel components like Image, Progress, Issues, or Text box and put them on the canvas. There is also a field for the Template Name and buttons

for actions like Save Draft and Preview Template.

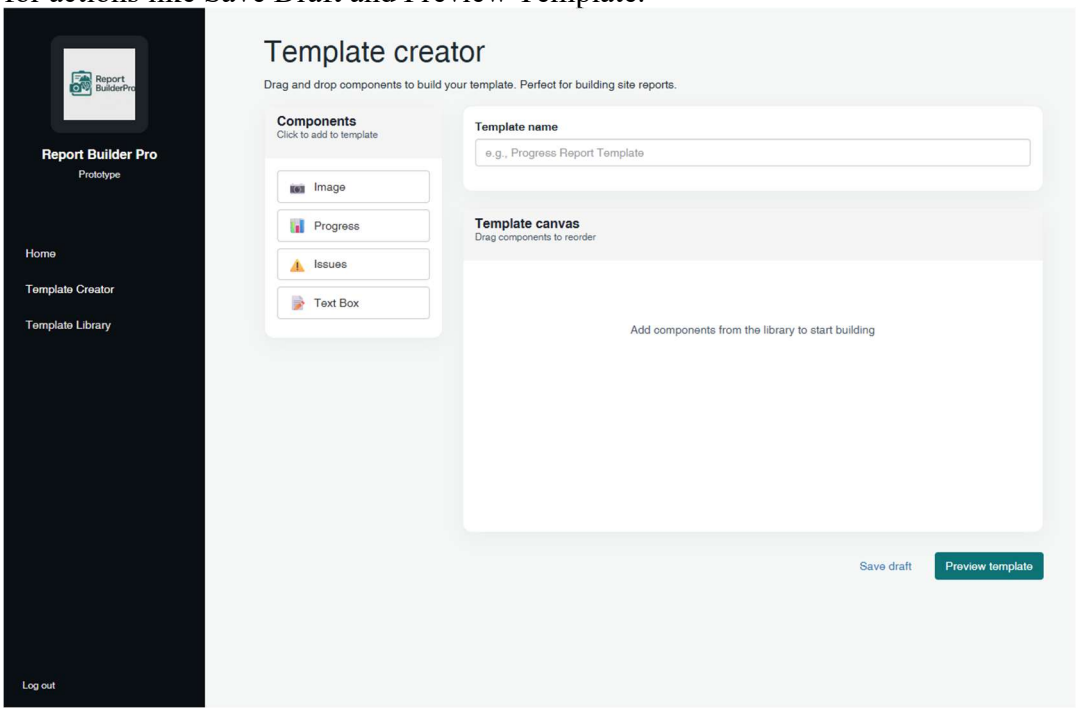


Figure 20 Template Creation page

```
// Add a new component to the template
const addComponent = (type: ComponentType) => {
  const newComponent: Component = {
    id: `comp-${Date.now()}`,
    type,
    data: {},
  };
  setTemplateComponents([...templateComponents, newComponent]);
};

// Handle drag n drop reordering of components
const handleDragEnd = (event: DragEndEvent) => {
  const { active, over } = event;
  if (!over || active.id === over.id) return;

  const oldIndex = templateComponents.findIndex((c) => c.id === active.id);
  const newIndex = templateComponents.findIndex((c) => c.id === over.id);
  const newComponents = [...templateComponents];
  const [moved] = newComponents.splice(oldIndex, 1);
  newComponents.splice(newIndex, 0, moved);
  setTemplateComponents(newComponents);
};
```

Figure 21 Dynamically add new component to canvas

This snippet illustrates how the Template Builder dynamically adds new components to the canvas and reorders them through drag-and-drop functionality, updating the component list in real time using React state management.

In the final implementation, these changes will be added to MongoDB via an API to enable saving and collaborative editing of templates.

Template Library

The Template Library is a central place where users can find and use pre-made templates quickly and easily. The templates are shown in a structured grid layout, which lets users see them, choose them, and change them as needed. This feature makes it easier to make reports by cutting down on repetitive work and making sure that all projects are the same.

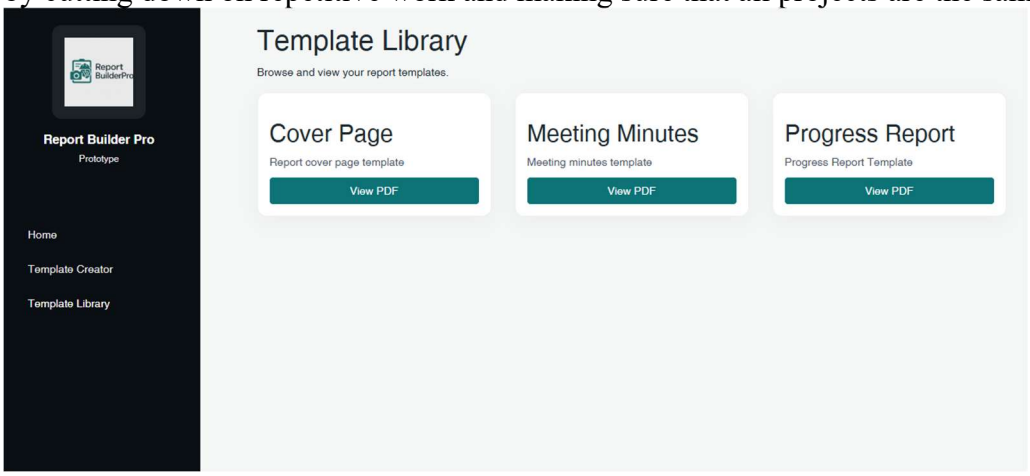


Figure 22 Template Library

```
// Fetch template list from MongoDB via backend API
useEffect(() => {
  const fetchTemplates = async () => {
    try {
      const response = await fetch(`${API_BASE_URL}/api/templates`);
      if (response.ok) {
        const data = await response.json();
        setTemplates(data);
      }
    } catch (error) {
      console.error('Failed to fetch templates:', error);
    } finally {
      setLoading(false);
    }
  };
  fetchTemplates();
<div className="row">
  {templates.map((template) => (
    <div key={template.id} className="col-sm-4">
      <div className="panel panel-default">
        <div className="panel-body">
          <h2>{template.title}</h2>
          <p className="text-muted">{template.description}</p>
          <button className="btn btn-rbp btn-block" onClick={() => openP
            View PDF
          </button>
        </div>
      </div>
    </div>
  ))}
</div>
```

Figure 23 Fetching Templates from MongoDB and Dynamic Render

These snippets shows how the Template Library retrieves templates from the backend using useEffect and renders them dynamically with Reacts map() function. Each template includes a title, description, and Base64-encoded PDF data for preview purposes. While this prototype approach enables quick testing, the final implementation will generate PDFs directly from the Template Creator and store only metadata in MongoDB. This exercise was helpful because it showed how useful PDF storage could be in the future, such as when you need to access files offline or keep them for a long time.

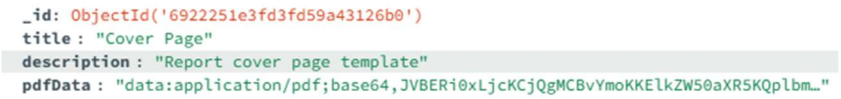


Figure 24 Sample of Template Data Store

This shows how templates are stored in MongoDB during the prototype phase, including Base64 encoded PDF data for previews. In the final implementation, PDFs will be generated from the Template Creator and only metadata will be stored for efficiency.

Accessibility

I did two different accessibility tests on the Prototype, using both the Arc Toolkit and Googles built in Lighthouse [31] . The app got a perfect score of 100 and alerts, which shows that it follows best practices for accessibility, like using semantic HTML, proper labelling, and colour contrast. In the early stages, this makes sure that the platform is on track to be follow usability and accessibility guidelines.

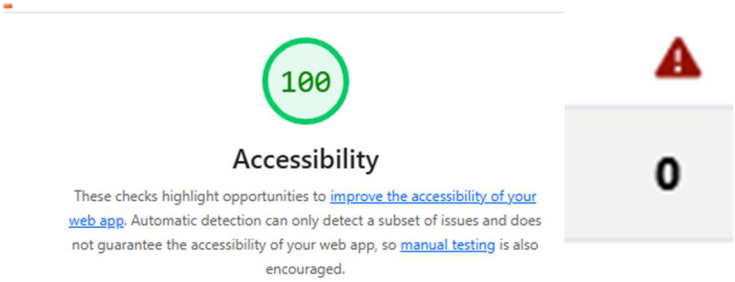


Figure 25 Accessibility Report Results

NLP Development

*Incomplete at time of submission

To investigate how NLP can be used in ReportBuilderPro I am working on a proof of concept that will help find problems with construction in reports that are mostly text-based. Tokenization, removing stop words, and normalizing are some of the methods that the prototype will use. A rule-based method will be used to find issues, using a dictionary of common construction terms (like "crack," "leak," and "hazard") and simple pattern matching.

When issues are found, they are put into logical groups like Safety, Quality, and Compliance, and the output is structured so that it is easy to understand. This first implementation shows that automated issue identification is possible and sets the stage for future improvements, such as machine learning models for context-aware detection and risk scoring.

In the next phase, the prototype will be improved by adding a bigger database of possible problems. This will make it easier to sort them into groups and open the door to more advanced machine learning features.

Scope Limitation

The current prototype focuses on establishing the foundational elements of the application rather than delivering full functionality. It includes a basic login system and a home screen featuring a static dashboard alongside a placeholder template creator. MongoDB integration has been implemented to store PDF files in Base64 format, supported by a temporary library for file handling. This version prioritises core connectivity, interface structure, and proof of concept, while advanced features such as complete template creation and dynamic dashboard

functionality are deferred to later development phases. As well as developing a proof of concept for NLP.

6.3 Results

Prototype Testing

To validate the core functionality outlined in the Section 5, we conducted a series of basic tests focused on the implemented features. These tests were designed to confirm connectivity, interface behaviour, and data handling within the current scope. Specifically, we verified.

Login

Ensured that user credentials are correctly retrieved from MongoDB and that authentication logic responds appropriately to valid and invalid inputs. It was also important to verify we could not sign in without the correct credentials.

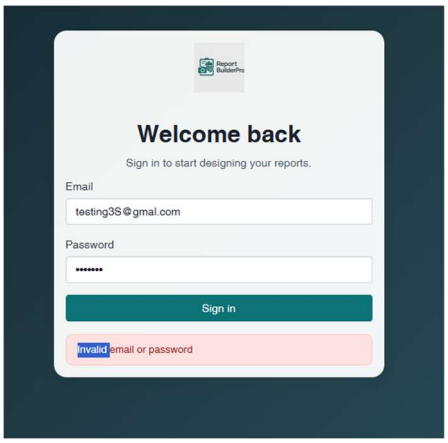


Figure 26 Attempt sign in with Incorrect Details

Template Builder

Tested drag-and-drop functionality for component reordering and the addition of new elements to the canvas.

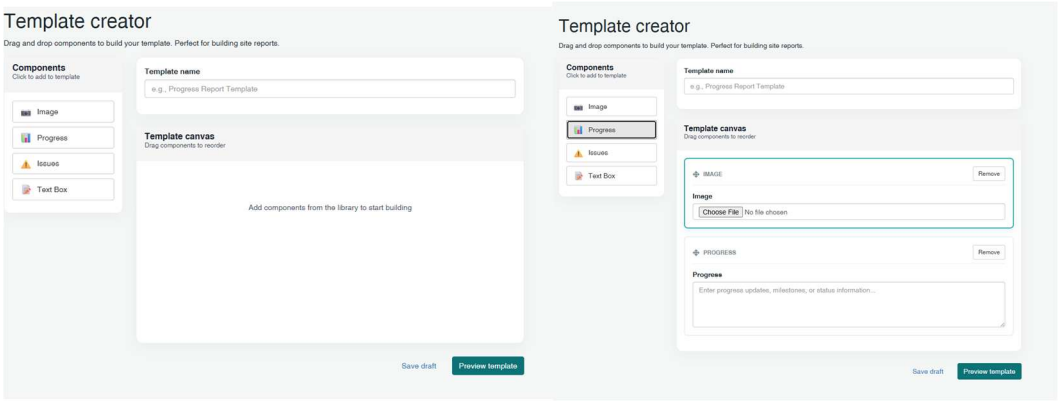


Figure 27 Displaying a blank and filled in canvas

Template Library

Confirmed that templates are fetched from the backend and rendered dynamically, including the ability to preview the template. This test also verified the MongoDB Integration as the template was successfully retrieved.

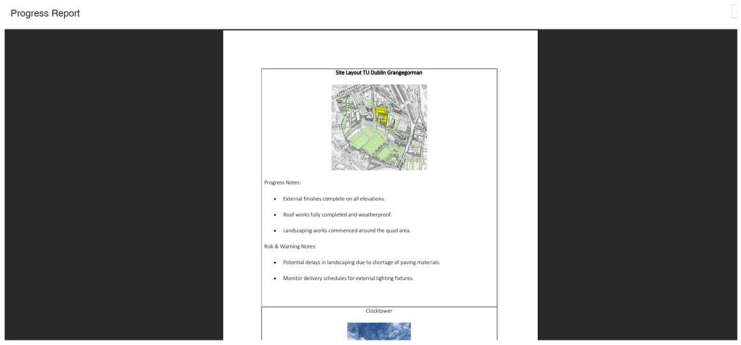


Figure 28 Sample of how a Template looks when opened

Accessibility

Conducting the accessibility audits provided valuable insights into how our designs align with WCAG guidelines and overall usability.

Prototype Development Summary Table

Task	Description	Result
Create GitHub Repository	Set up version control and collaboration environment	Successful
Set Up Development Environment	Installed React, Node.js, MongoDB, and configured project structure	Successful
Implement Login Screen	Built a simple authentication interface	Successful
Connect Login to MongoDB	Integrated backend logic for user credential validation	Successful
Dashboard Layout	Created a basic dashboard with navigation and placeholders	Successful
Template Library	Added initial template preview functionality	Successful
Accessibility Audit	Conducted WCAG compliance checks on prototype	Successful
Usability Testing	Tested navigation and layout for intuitiveness	Successful

6.4 Evaluation

This section evaluates the outcomes of the development phase using the approaches outlined in Section 5. The evaluation focuses on accessibility compliance, usability, and performance to ensure the solution meets project objectives and user needs.

Prototype Evaluation

Evaluation Method	Criteria/Guideline	Findings	Outcome	Recommendations

Accessibility Audit	WCAG 2.1	Minor colour contrast issues detected	Partially compliant	Adjust colour palette for better contrast
Usability Testing	Navigation & Layout	Navigation intuitive	Mostly positive	Improve clarity of form labels
Performance Check	Load Time < 3s	Average load time: 2.8s	Meets target	N/A

Overall, the evaluation confirms strong alignment with the accessibility and usability principles defined in our design guidelines and system analysis for Report Builder Pro. The findings demonstrate that the implemented features adhere to WCAG standards and support the user-centric approach outlined during the analysis phase. Minor areas for improvement such as colour contrast and form label clarity will further strengthen compliance and enhance the user experience. Performance meets expectations for the current scope, validating the design decisions made during system planning. As development progresses through future iterations of ReportBuilderPro prototypes, the project will continue to follow the established design guidelines and accessibility standards outlined in the system analysis.

6.5 Conclusions

The initial prototype of ReportBuilderPro successfully established the foundational elements of the application, including user authentication, dashboard navigation, and template placeholders. These features demonstrate connectivity with the MongoDB database and provide a functional starting point for future development.

The evaluation showed that the prototype follows the system analysis and design guidelines for accessibility and usability. There were some small issues, like colour contrast and form label clarity, but these findings show how important it is to keep refining the design in order to fully meet WCAG standards, provide the best user experience, and let me fix them as we continue with development.

The prototype shows that the proposed architecture and design approach will work in general. It lays a strong foundation for adding more advanced features in future versions, like secure authentication, dynamic data handling, and better template management. As development goes on, ReportBuilderPro will conform to the rules and evaluation methods that have already been outlined to make sure that the design is consistent, high-quality, and focused on the user.

7. Issues and Future Work

7.1 Introduction

This section outlines the issues encountered during the development of ReportBuilderPro and identifies potential risks that may affect the completion of the system. It also presents strategies for addressing these challenges and ensuring that the final implementation meets the objectives defined in the system analysis and design phases.

In addition, this section provides a forward-looking plan for future development, detailing how remaining tasks will be prioritised and scheduled. By addressing these issues systematically, the project aims to deliver a secure, accessible, and fully functional reporting tool within the available timeframe.

7.2 Issues and Risks

7.2.1 Managing Time Constraints and Workload

One of the main challenges for this project is balancing development with other academic commitments, including modules and exams. While the Feature-Driven Development (FDD) approach has helped break the project into smaller, manageable tasks, the overall scope remains significant. Clear prototypes and phased development have provided structure, but achieving the desired standard will require disciplined time management and prioritisation.

7.2.2 NLP Training and Data Availability

Implementing natural language processing (NLP) for automated report generation presents several challenges. The most significant is the need for a large, domain-specific dataset to train the model effectively. Collecting and preparing this data is time-consuming and may not be fully achievable within the project timeline.

Training an NLP model also requires computational resources and careful preprocessing to ensure accuracy and relevance. Without sufficient data, the model may produce incomplete or inaccurate outputs, limiting its reliability for real-world use.

Despite these constraints, the initial implementation will demonstrate the concept and provide a functional starting point for automated reporting. While it will not be production-ready, it will validate the approach and lay the groundwork for future iterations, where more comprehensive datasets and refined algorithms can be incorporated.

7.3 Plans and Future Work

In order to complete ReportBuilderPro within the available timeframe, development will build on the current web application, which already includes core functionality such as user authentication, dashboard navigation, and template placeholders. The next steps will focus on integrating a simplified mobile application for field reporting as well as implementing and training the NLP component for automated report generation.

Addressing issues such as colour contrast adjustments, form label clarity, and security enhancements will remain a priority. Dynamic data handling will replace hardcoded elements, ensuring real time updates from the MongoDB backend.

Key risks such as time constraints and technical challenges will be mitigated through structured scheduling, feature-driven development, and continuous evaluation. This approach

ensures that essential features such as mobile integration, NLP integration, and security are delivered first, with refinements added as time allows.

7.3.1 Project Plan with GANTT Chart

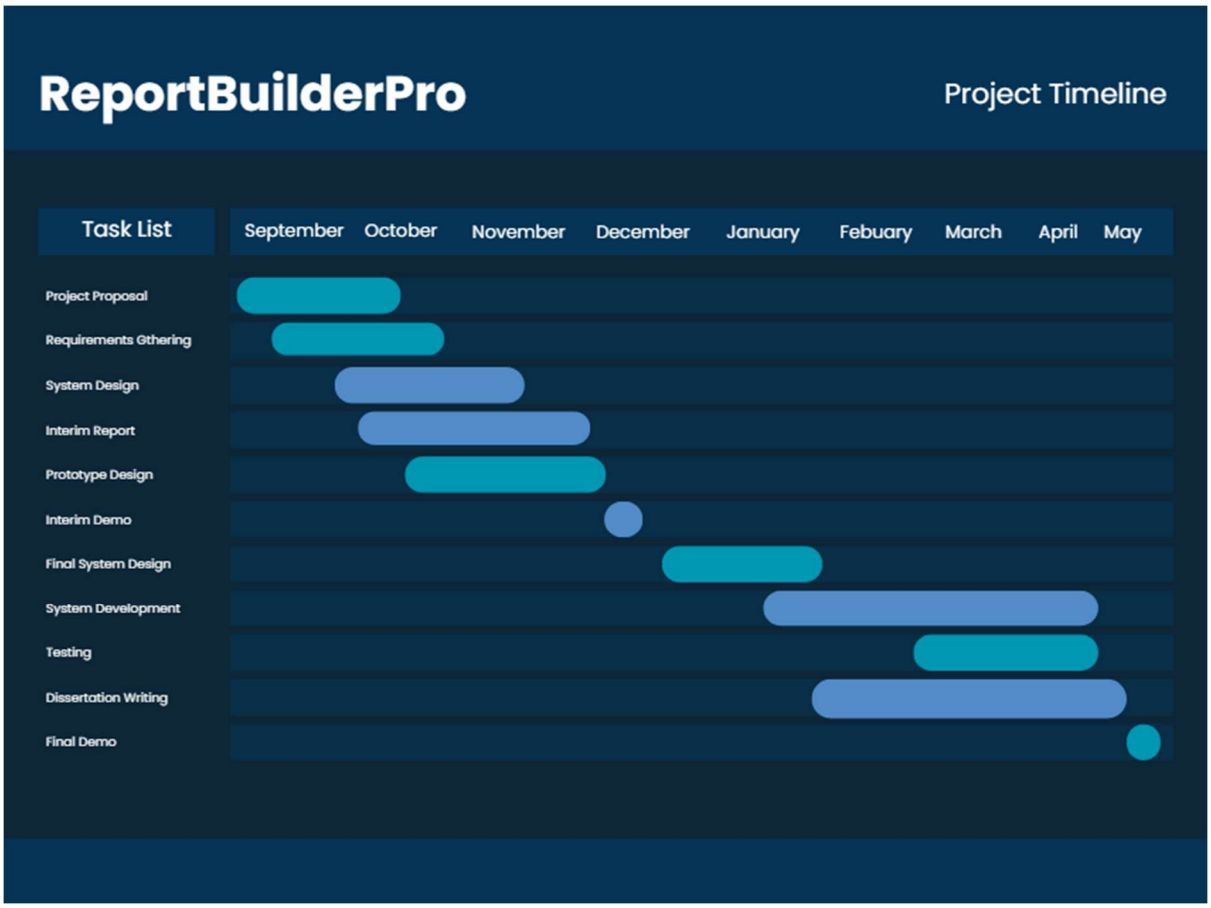


Figure 29 Gannt Chart Outlining Project Timeline

References

[1] ‘Spotlight on Build Digital: Driving Innovation in Irish Construction and Built Environment Industries’, gov.ie. Accessed: Nov. 21, 2025. [Online]. Available: https://www.gov.ie/en/department-of-public-expenditure-infrastructure-public-service-reform-and-digitalisation/spotlight-on-build-digital-driving-innovation-in-irish-construction-and-built-environment-industries/

[2] ‘(PDF) The Challenges of Standardization of Products and Processes in Construction’, in *ResearchGate*, doi: 10.13140/2.1.3993.7600.

[3] D. Shamsollahi, ‘Construction Progress Monitoring and Reporting using Digital Images and Computer Vision Techniques - A Review’. [Online]. Available: https://www.iaarc.org/publications/2022_proceedings_of_the_39th_isarc_bogota_colombia/construction_progress_monitoring_and_reporting_using_digital_images_and_computer_vision_techniques_a_review.html

[4] Y. Adebayo, P. Udoh, X. B. Kamudyariwa, and O. A. Osobajo, ‘Artificial Intelligence in Construction Project Management: A Structured Literature Review of Its Evolution in Application and Future Trends’, *Digital*, vol. 5, no. 3, p. 26, Sept. 2025, doi: 10.3390/digital5030026.

[5] ‘Layout 1’. Accessed: Nov. 21, 2025. [Online]. Available: https://scsi.ie/wp-content/uploads/2025/08/SurveyorsJournal_August2025_web.pdf

[6] S. of C. S. Ireland, ‘Will AI help quantity surveying?’, *Surv. J.*, vol. 25, no. 3, pp. 20–22, Aug. 2025.

[7] D. Vararean-Cochisa and E.-L. Crisan, ‘The digital transformation of the construction industry: a review’, *IIM Ranchi J. Manag. Stud.*, vol. 4, no. 1, pp. 3–16, Jan. 2025, doi: 10.1108/IRJMS-04-2024-0035.

[8] E. Ireland, ‘EY report highlights critical role of innovation to ensure a productive and sustainable construction sector in Ireland’, EY report highlights critical role of innovation to ensure a productive and sustainable construction sector in Ireland. [Online]. Available: https://www.ey.com/en_ie/newsroom/2023/02/ey-report-highlights-critical-role-of-innovation-to-ensure-a-productive-and-sustainable-construction-sector-in-ireland

[9] ‘Home’, The National Planning Framework. Accessed: Nov. 21, 2025. [Online]. Available: https://www.npf.ie/

[10] ‘Government publishes updated National Development Plan’, gov.ie. Accessed: Nov. 21, 2025. [Online]. Available: https://www.gov.ie/en/department-of-the-taoiseach/press-releases/government-publishes-updated-national-development-plan/

[11] A. Naik, ‘The Front-End Dilemma: How to Choose the Perfect TECHNOLOGY for Your Application.’, *J. Comput. Sci. Technol. Stud.*, vol. 6, no. 1, pp. 211–216, Mar. 2024, doi: 10.32996/jcsts.2024.6.1.24.

[12] ‘The Invisible Technology How Backend Systems Shape Our Digital Lives | Request PDF’, ResearchGate. Accessed: Nov. 21, 2025. [Online]. Available: https://www.researchgate.net/publication/391980564_The_Invisible_Technology_How_Backend_Systems_Shape_Our_Digital_Lives

[13] M. Rathore and S. S. Bagui, ‘MongoDB: Meeting the Dynamic Needs of Modern Applications’, *Encyclopedia*, vol. 4, no. 4, pp. 1433–1453, Dec. 2024, doi: 10.3390/encyclopedia4040093.

[14] V. Hemming, M. A. Burgman, A. M. Hanea, M. F. McBride, and B. C. Wintle, ‘A practical guide to structured expert elicitation using the IDEA protocol’, *Methods Ecol. Evol.*, vol. 9, no. 1, pp. 169–180, 2018, doi: 10.1111/2041-210X.12857.

[15] ‘Feature Driven Development (FDD) and Agile Modeling’. Accessed: Nov. 03, 2025. [Online]. Available: https://agilemodeling.com/essays/fdd.htm

[16] ‘TU Dublin - Library Catalogue’. Accessed: Nov. 21, 2025. [Online]. Available: https://library.tudublin.ie/#
[17] J. Hew and M. Llorens Salvador, ‘Anseo! -: A modern approach to tracking student attendancein university Final project report’, TU Dublin, Dublin, 2023.
[18] B. McCarthy and J. Carswell, ‘Deep: Final project report’, TU Dublin, Dublin, 2023.
[19] S. Lewellen, ‘Identifying Key Stakeholders as Part of Requirements Elicitation in Software Ecosystems’, in *Proceedings of the 24th ACM International Systems and Software Product Line Conference - Volume B*, in SPLC ’20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 88–95. doi: 10.1145/3382026.3431249.
[20] ‘(PDF) The Significance of Use Case Diagrams in Software Development Keywords Use Case Diagrams Software Development Requirements Elicitation System Design Stakeholder Communication Functional Requirements Visual Modeling Tools Iterative Development User-Centered Design Software Engineering Methodologies’, ResearchGate. Accessed: Nov. 14, 2025. [Online]. Available: https://www.researchgate.net/publication/387903437_The_Significance_of_Use_Case_Diagrams_in_Software_Development_Keywords_Use_Case_Diagrams_Software_Development_Requirements_Elicitation_System_Design_Stakeholder_Communication_Functional_Requirements_V
[21] ‘ISO 9241-110:2020(en), Ergonomics of human-system interaction — Part 110: Interaction principles’. Accessed: Nov. 21, 2025. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:9241:-110:ed-2:v1:en
[22] E. Wong, ‘Shneiderman’s Eight Golden Rules Will Help You Design Better Interfaces’, The Interaction Design Foundation. Accessed: Nov. 14, 2025. [Online]. Available: https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces
[23] ‘Web Content Accessibility Guidelines (WCAG) 2.2’. Accessed: Nov. 15, 2025. [Online]. Available: https://www.w3.org/TR/WCAG22/
[24] Asana, ‘What Is Agile Methodology? (A Beginner’s Guide) [2025] • Asana’, Asana. Accessed: Nov. 14, 2025. [Online]. Available: https://asana.com/resources/agile-methodology
[25] ‘What is Scrum? | Scrum.org’. Accessed: Nov. 21, 2025. [Online]. Available: https://www.scrum.org/resources/what-scrum-module
[26] ‘HTTP: Hypertext Transfer Protocol | MDN’, MDN Web Docs. Accessed: Nov. 21, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP
[27] ‘What is an Entity Relationship Diagram? | IBM’. Accessed: Nov. 21, 2025. [Online]. Available: https://www.ibm.com/think/topics/entity-relationship-diagram
[28] ‘Modular Software Design: Principles, Polymorphism, and Patterns’, in *ResearchGate*. doi: 10.1007/978-1-4842-4143-1_1.
[29] M. Aniche, *Effective Software Testing: A Developer’s Guide*. Simon and Schuster, 2022.
[30] K. W. Prima, ‘Balancing Data Security: A Comparative Review of Hashing and Non-Hashing Approaches in Data Storage | Journal of Embedded Systems, Security and Intelligent Systems’, Oct. 2025, Accessed: Nov. 14, 2025. [Online]. Available: https://journal.unm.ac.id/index.php/JESSI/article/view/9257
[31] ‘Introduction to Lighthouse’, Chrome for Developers. Accessed: Nov. 20, 2025. [Online]. Available: https://developer.chrome.com/docs/lighthouse/overview

A) Appendix A: System Model and Analysis

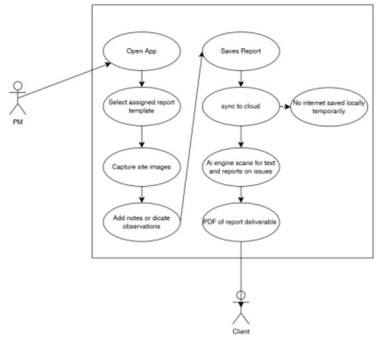


Figure 30 Sample Mobile use Case

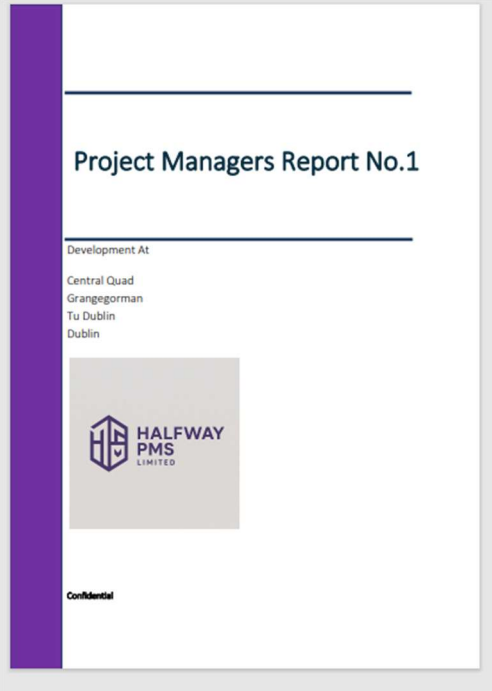
Questionnaire Stakeholders
Target Group: Quantity Surveyors, Site Engineers
Justification: They are frequent users of reporting tools.
Sample Questions:

1. How often do you submit site reports?
2. Do you use mobile apps for reporting?
3. What device do you use most?
4. Do you work in areas with poor connectivity?
5. Would offline access help?
6. How do you currently tag images?
7. Would AI tagging save time?
8. Do you use templates?
9. What format do you prefer for reports?
10. What features would you like in a new system?

Figure 31Initial Questionnaire sample

Stakeholder	Posistve Impact	Negative Impact
Quantity Surveyors	Easier reporting less paperwork	Learning new tech
Project Manager	Real time insigts easier coordination	Initial learning + setup
Admin	Standardised layout	Managing the data

Figure 32 Some Possible Impacts on the current workflow



Project Managers Report No.1

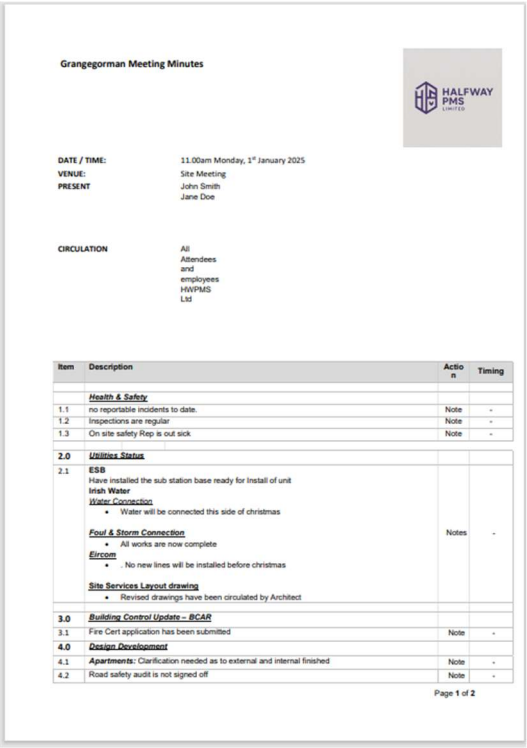
Development At

Central Quad
Grangegorman
Tu Dublin
Dublin

HALFWAY PMS LIMITED

Confidential

Figure 33 Example Cover Page for Report



Grangegorman Meeting Minutes

HALFWAY PMS LIMITED

DATE / TIME: 11.00am Monday, 1st January 2025
VENUE: Site Meeting
PRESENT: John Smith, Jane Doe

CIRCULATION: All Attendees and employees HWPMS Ltd

Item	Description	Action	Timing
	Health & Safety		
1.1	no reportable incidents to date.	Note	-
1.2	Inspections are regular	Note	-
1.3	On site safety Rep is out sick	Note	-
2.0	***Utilities Status***		
2.1	**ESB** Have installed the sub station base ready for Install of unit **Irish Water** *Water Connection* • Water will be connected this side of christmas ***Foul & Storm Connection*** • All works are now complete ***Eircom*** • . No new lines will be installed before christmas **Site Services Layout drawing** • Revised drawings have been circulated by Architect	Notes	-
3.0	***Building Control Update – BCAR***		
3.1	Fire Cert application has been submitted	Note	-
4.0	***Design Development***		
4.1	***Apartments:*** Clarification needed as to external and internal finished	Note	-
4.2	Road safety audit is not signed off	Note	-

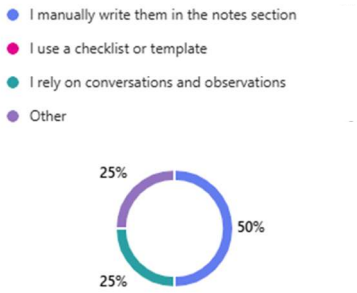
Page 1 of 2

Figure 34 Meeting Minutes Sample

Elicitation Results

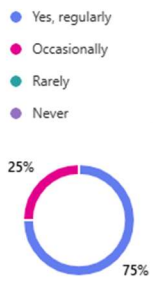
1. How do you currently identify risks or delays in your site reports?

More details



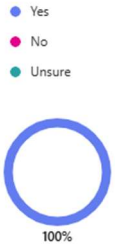
2. Do you feel that risks or delays are sometimes missed or underreported?

More details



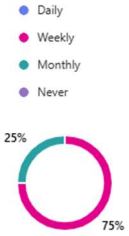
3. If Software existed that could scan Reports and find potential issues ie Risks, delays material shortages...

More details



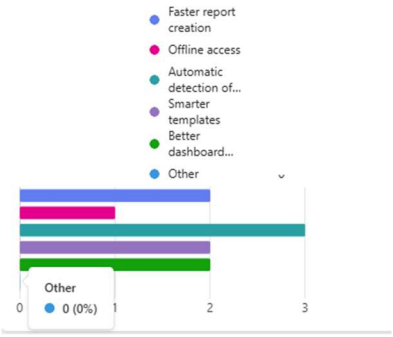
4. How often do you work in areas with poor or no internet connection when not in the office?

More details



7. What would make your reporting workflow easier? *(Tick all that apply)*

More details



9. Would a system that automatically flags risks and delays from report text and works offline help...

More details

B) Appendix B: Design

The system uses a modular architecture with a React-based front end and Node.js backend. Components are organised by functionality,

```
{
  id: 2,
  jobId: '2024-045',
  title: 'Documentation Incomplete',
  description: 'Progress report missing required signatures',
  severity: 'medium',
  category: 'Compliance',
  date: '5 hours ago',
},
```

Figure 35 Sample Data Design AI Issue

```
const uploadImage = (id: string, event: React.ChangeEvent<HTMLInputElement>)
  const file = event.target.files?.[0];
  if (file) {
    const reader = new FileReader();
    reader.onload = (e) => {
      const imageData = e.target?.result as string;
      updateComponent(id, 'image', imageData);
    };
    reader.readAsDataURL(file);
  }
};
```

Figure 36 Image Upload Handle

Figure 37 Logo

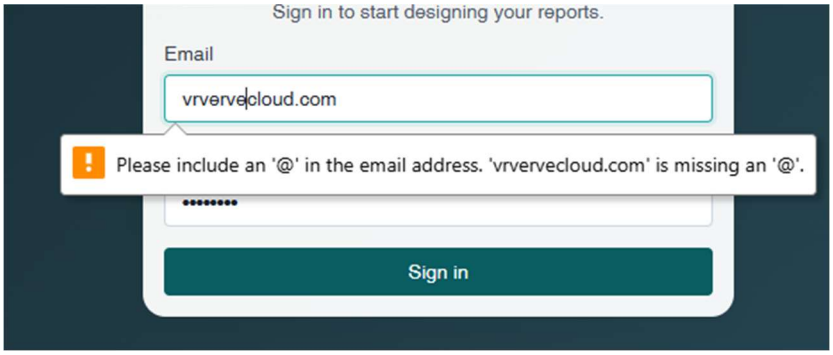


Figure 38 Reminder to add the @

C) Appendix C: Prompts Used with M365 Copilot

Copilot was utilised throughout the documentation process to refine grammar, maintain a professional and academic tone, and suggest alternative phrasing where necessary.

Provided an excel I drafted with a comparison of large number of reporting applications

"Create a feature comparison matrix for all apps in my Excel sheet, highlighting unique features and common gaps. And possible suggestions for improvement possible with an AI element"

"Evaluate how the listed apps perform on mobile versus desktop. Highlight usability issues and suggest improvements for a new app to ensure seamless cross-platform experience."

"I have designed a site reporting system that includes mobile data capture, photo annotation, voice-to-text reporting, automated report generation, and integration with project management tools. The system should allow offline functionality, cloud storage, and role-based access for site engineers, project managers, and clients. If I were to visualise the entire workflow, how might it look"

"Help me find simple software for creating a Gantt chart for my college dissertation. I want to build the chart myself without using complex tools."

"I want to use NLP to analyse site reports and automatically identify potential issues, risks, and delays. I already know this involves entity recognition and pattern detection, but can you confirm the best approach for extracting risk-related keywords and suggest one technique for flagging delays based on text patterns?"

"Can this process work effectively when the input data is partially unstructured, such as free-text notes or inconsistent formatting?"

"Here is the architecture for Report Builder Pro: [paste architecture details]. Can you help me break this down into a clear visual structure so I can create a system architecture diagram in draw.io?"

D) Appendix D: Additional Code Samples

Presentation Layer

```
<div className="panel-body">
  {templateComponents.length === 0 ? (
    <div className="canvas-empty">
      <p className="text-muted">Add components from the library to s
    </div>
  ) : (
    <DndContext sensors={sensors} collisionDetection={closestCenter}
      <SortableContext items={templateComponents.map((c) => c.id)} s
        {templateComponents.map((component) => (
          <ComponentItem
            key={component.id}
            component={component}
            onUpdate={updateComponent}
            onRemove={removeComponent}
            onImageUpload={uploadImage}
          />
        ))}
      </SortableContext>
    </DndContext>
  )}
</div>
```

Figure 39 UI logic for template building